НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА СИСТЕМНОГО ПРОГРАМУВАННЯ І
СПЕЦІАЛІЗОВАНИХ КОМП'ЮТЕРНИХ СИСТЕМ

«На правах рукопису»
УДК 004.05

«До захисту допущено»

Завідувач кафедри СПСКС

_____ В.П.Тарасенко_
(підпис) (ініціали, прізвище)

"___"_____2018р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 123 Комп'ютерна інженерія
Комп'ютерні системи та компоненти

на тему МЕТОД СКОРОЧЕННЯ ЧАСУ САМОДІАГНОСТУВАННЯ У БАГАТОПРОЦЕСОРНИХ СИСТЕМАХ

Виконала: студентка ІІ курсу, групи КВ-71мп
(шифр групи)

Довганюк Анна Олегівна
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник д.т.н., доц. Романкевич В.О. __ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент_____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра системного програмування та спеціалізованих комп'ютерних систем

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою

Спеціальність (спеціалізація) – 123 «Комп'ютерна інженерія»

«Комп'ютерні системи та компоненти»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Тарасенко В.П.

«___»_____2018 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Довганюк Анні Олегівні

1. Тема дисертації «Метод скорочення часу самодіагностування у багатопроцесорних системах», науковий керівник дисертації Романкевич Віталій Олексійович, д.т.н., доцент, затверджені наказом по університету від «30» жовтня 2018 р. №4030-с

2. Термін подання студентом дисертації «7» грудня 2018 р.

3. Об'єкт дослідження: багатопроцесорні системи, зокрема відмовостійкі.

4. Предмет дослідження: процеси самодіагностування багатопроцесорних систем.

5. Перелік завдань, які потрібно розробити:
- провести аналіз методів самодіагностування багатопроцесорних систем;
- дослідити методи самодіагностування, які можуть бути використані при розробці методу зменшення часу самодіагностування у багатопроцесорних системах;
- розробити метод зменшення часу самодіагностування у багатопроцесорних системах;
- розробити програмне забезпечення для моделювання роботи запропонованого методу;

– провести аналіз результатів, отриманих при моделюванні роботи методу.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу:
   – презентація до магістерської дисертації.

7. Орієнтовний перелік публікацій:
   – Тези доповіді “Про самодіагностування багатопроцесорних систем”;
   – Тези доповіді “Метод зменшення часу самодіагностування у багатопроцесорних системах”.

8. Дата видачі завдання «04» жовтня 2017 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Ґрунтовне ознайомлення з предметною галуззю	15.09.2017	
2.	Визначення структури магістерської дисертації; вивчення літератури, пошук додаткової літератури, патентний пошук	20.12.2017	
3.	Робота над першим розділом магістерської дисертації; проведення наукового дослідження	1.03.2018	
4.	Проведення наукового дослідження; робота над другим розділом магістерської дисертації; розроблення програмного забезпечення	31.03.2018	
5.	Проведення наукового дослідження; робота над статтею за результатами наукового дослідження	13.04.2018	
6.	Проведення наукового дослідження; робота над третім розділом магістерської дисертації	15.06.2018	
7.	Завершення роботи над основною частиною магістерської дисертації; підготовка ілюстративного матеріалу; підготовка матеріалів доповіді на конференції ПМК-2018	25.10.2018	
8.	Оформлення текстової і графічної частини магістерської дисертації	20.11.2018	

Студент А.О. Довганюк

Науковий керівник дисертації В.О. Романкевич

РЕФЕРАТ

Актуальність теми. Необхідність у високопродуктивних засобах обробки інформації призвела до створення розподілених обчислювальних систем (ОС). Проте технологічні системи уразливі до несправностей, і неправильна робота елементів системи знижують її продуктивність та можуть призвести до виходу системи з ладу. Як наслідок, виявлення та обробка несправностей відіграють все більшу роль у сучасних технологіях, оскільки при взаємодії великої кількості компонентів, несправність в одному із них може призвести до несправності всієї системи.

Вирішення цієї проблеми включає в себе розробку програм, що моделюють виникнення несправностей, генерацію оптимальних наборів діагностичних тестів, а також створення самодіагностованих інформаційно-обчислювальних систем (СІОС).

Організація самодіагностування у багатопроцесорних системах, а також здатність знаходження та локалізації несправного процесора, суттєво знижують час, який система витрачає на відновлення. Надійність таких систем визначається не тільки відсутністю збоїв у їх роботі, але й здатністю швидкого відновлення роботи при виникненні відмов компонентів. У процесі розробки самодіагностованої системи необхідно враховувати можливість виникнення часової та/або алгоритмічної надлишковості в результаті забезпечення високої надійності.

Об’єктом дослідження є багатопроцесорні системи, зокрема відмовостійкі.

Предметом дослідження є процеси самодіагностування багатопроцесорних систем.

Мета роботи: скорочення часу самодіагностування у багатопроцесорних системах.

Методи дослідження. В роботі використовуються методи дискретної математики, методи самодіагностування багатопроцесорних систем.

Наукова новизна полягає в наступному:

Запропоновано метод організації самодіагностування багатопроцесорних систем та виконано оцінку кількості взаємоперевірок, дано верхня та нижня границі.

Практична цінність отриманих в роботі результатів полягає в тому, що запропонований метод дозволяє зменшити час самодіагностування багатопроцесорних систем шляхом зменшення кількості тестових перевірок і тим самим підвищує продуктивність системи.

Апробація роботи. Основні положення і результати роботи були представлені та обговорювались на XI науковій конференції молодих вчених «Прикладна математика та комп'ютинг» ПМК-2018-2 (Київ, 14-16 листопада 2018 р.) та 20-й Міжнародній науково-технічної конференції SAIT 2018, Київ, 21 – 24 травня 2018 р.

Структура та обсяг роботи.

У вступі надано загальну характеристику роботи, обґрунтовано актуальність напрямку досліджень, сформульовано мету і задачі досліджень, показано наукову новизну отриманих результатів і практичну цінність роботи.

У першому розділі розглянуто задачу самодіагностування, розглянуто теоретичні засади, які були взяті за основу дослідження.

У другому розділі обрано модель несправностей, надано характеристику та порівняння існуючих методів самодіагностування багатопроцесорних систем.

У третьому розділі сформульовано основні положення методу зменшення часу самодіагностування у багатопроцесорних системах, обґрунтовано вибір технічних засобів для створення програмної реалізації запропонованого методу, надано опис структури програмної моделі та принцип її роботи.

У четвертому розділі надано ґрунтовний аналіз роботи методу скорочення часу самодіагностування у багатопроцесорних системах.

У висновках узагальнені результати дослідження.

Магістерська дисертація виконана на 84 аркушах, містить 4 додатки та посилання на список використаних літературних джерел з 13 найменувань. У роботі наведено 23 рисунка та 3 таблиці.

Ключові слова: самодіагностування, багатопроцесорні системи.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

АМН	Асиметрична модель несправності
ВБС	Відмовостійка багатопроцесорна система
ДМ	Діагностична модель
ДС	Діагноз системи
ОД	Об'єкт діагностування
ОС	Обчислювальна система
ПМЧ-модель	(Prepatata, Metze, Chien) – модель несправностей, розроблена трьома авторами, на неї спирається теоретична база даної роботи.
СБС	Самодіагностована багатопроцесорна система
СІОС	Самодіагностована ІОС
СМН	Симетрична модель несправності
ТС	Технічна система

ВСТУП

Необхідність у високопродуктивних засобах обробки інформації призвела до створення розподілених обчислювальних систем (ОС). Проте технологічні системи уразливі до несправностей, і неправильна робота елементів системи знижують її продуктивність та можуть призвести до виходу системи з ладу. Як наслідок, виявлення та обробка несправностей відіграють все більшу роль у сучасних технологіях, оскільки при взаємодії великої кількості компонентів, несправність в одному із них може призвести до несправності всієї системи [2].

Вирішення цієї проблеми включає в себе розробку програм, що моделюють виникнення несправностей, генерацію оптимальних наборів діагностичних тестів, а також створення самодіагностованих інформаційно-обчислювальних систем (СІОС).

Організація самодіагностування у багатопроцесорних системах, а також здатність знаходження та локалізації несправного процесора, суттєво знижують час, який система витрачає на відновлення. Надійність таких систем визначається не тільки відсутністю збоїв у їх роботі, але й здатністю швидкого відновлення роботи при виникненні відмов компонентів. У процесі розробки самодіагностованої системи необхідно враховувати можливість виникнення часової та/або алгоритмічної надлишковості в результаті забезпечення високої надійності [3].

Метою даного дослідження є оптимізація роботи процесу самодіагностування багатопроцесорної системи, а саме: зменшення часу, який система витрачає на визначення власного стану.

В даній магістерській дисертації запропоновано метод зменшення кількості перевірок, необхідних для визначення стану системи, що дозволяє скоротити час самодіагностування багатопроцесорних систем.

1. АКТУАЛЬНІСТЬ ТЕМИ ДОСЛІДЖЕННЯ ТА АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ОРГАНІЗАЦІЇ САМОДІАГНОСТУВАННЯ

1.1. Актуальність теми дослідження

Проблема виявлення несправних елементів у наборі взаємопов'язаних процесорів набуває все більшого значення у зв’язку з останніми розробками в області розподілених і паралельних обчислень. Проте при розробці таких систем необхідною вимогою має бути забезпечення високої надійності системи. Навіть у випадку відмови деякої кількості компонентів система має залишатися роботоздатною. В літературі для цього застосовується поняття «відмовостійкость». Під цим поняттям розуміють здатність системи продовжувати свою роботи навість при відмові її компонентів [1].

Забезпечення цієї властивості є однією з найважливіших вимог до багатопроцесорних систем. Починаючи 70-х років в цій області було проведено безліч досліджень. І вони все ще залишаються актуальними, оскільки з розвитком комп’ютерної галузі створюються все більш складні системи, як за кількістю компонентів, так і за функціями, які вони виконують. При організації процесу тестування системи необхідно оцінити необхідність забезпечення автономності системи в цілому, тобто можливість визначення системою власного технічного стану. У літературі описаний процес називається самодіагностуванням. У загальному випадку діагностичним засобам системи необхідно забезпечувати локалізацію компонентів, що відмовили, на різних рівнях роботи системи [2]. Це означає, що після процесу самодіагностування несправні модулі прибираються з подальшої роботи, а система продовжує своє функціонування за рахунок перерозподілення задач несправного модуля/модулей між працюючими компонентами системи. Це забезпечує роботу системи навіть при виникненні часткових відмов апаратури та збоїв.

1.2. Задача самодіагностування

Розуміння зміни процесів в системі при виході з ладу її компонентів є ключем до створення реалістичних припущень та розробки алгоритмів для виявлення та заміни несправних модулів. Проте, будуючи модель динамічного процесу для контролю за його поведінкою, завжди існує певна невідповідність між очікуваними та реальними моделями поведінки, оскільки деякими ефектами нехтують для спрощення моделювання, деякі параметри мають певне відхилення, коли порівнюються між деякими одиницями того ж компоненту, помилки в параметрах (або в структурі) моделі вносяться в процесі калібрування моделі і т. д.

Ці помилки моделювання вносять певну невизначеність у модель. Зазвичай ця невизначеність може бути обмежена і включена в модель виявлення несправностей. Існує декілька способів розгляду невизначеності, пов'язаної з моделлю, залежно від того, де вона знаходиться в параметрах (структурованих) або в структурі моделі (неструктурованих) [2]. У літературі алгоритм діагностики несправностей, здатний виконувати невизначеність, називається надійним. Надійність алгоритму – це ступінь чутливості до несправностей в порівнянні зі ступенем чутливості до невизначеності [1].

Ретельне вивчення характеристик несправних процесорів призвело до розробки різних моделей несправностей. Три типи моделей несправностей були широко вивчені впродовж останніх десятиліть: моделі недійсності, моделі порівняння та імовірнісні моделі. У моделях недійсності, таких як ПМЧ-модель, система розкладається на безліч чітко визначених вузлів обробки, і для кожного вузла застосовується один або декілька тестів [2]. Алгоритм діагностики несправностей використовується для вирішення проблеми точного виявлення несправностей у системі з урахуванням набору результатів тестування. Зіставлення моделей, з іншого боку, припускає, що

завдання покладаються на пари вузлів. Виходячи з угод та розбіжностей між парами вузлів, набір помилкових процесорів може бути правильно ідентифікований. Обидві моделі недійсності та порівняння передбачають обмежений стан несправності, тобто найгірший варіант поведінки завжди приймається в спробі гарантувати певний рівень діагнозу. Як альтернативу, були розроблені ймовірнісні моделі. За цими моделями розглядаються лише набори несправностей, що мають незначну ймовірність виникнення.

Розглянемо Залежно від типів несправностей розрізняють симетричні (рис.1.1а) та асиметричні (рис.1.1б) моделі несправностей (СМН/АМН) [3].

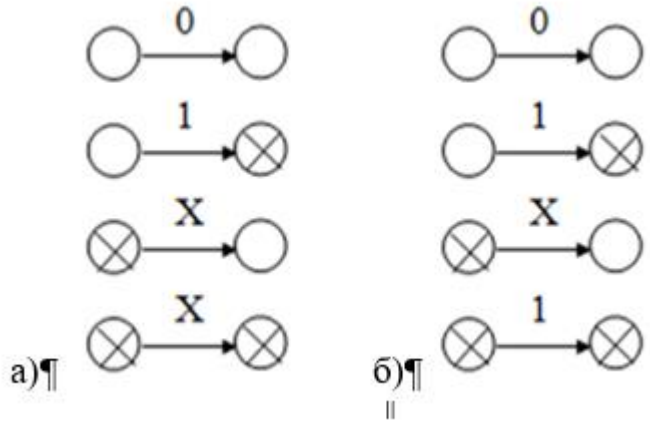


Рисунок 1.1 – Симетричні та асиметричні моделі несправностей

Для рис.1.1 діють наступні позначення:

◯ - справний модуль,

⊗ - постійно несправний модуль,

⊖ - модуль з несправністю, що перемежовується.

Модель несправностей, що зображена на рис.1.1а у літературі називається моделлю Препарата-Метца-Чена, вона запропонована у кінці 70-х років і досі залишається найбільш поширеною у дослідженнях проблем організації самодіагностування багатопроцесорних систем.

Модель на рис.1.1б є окремим випадком моделі Препарат-Метца-Чена і відрізняється неузгодженністю між очікуваним та дійсним результатом діагностування, коли несправний модуль є модулем, що тестується. Ця модель називається моделлю Барсі-Грандоні-Майстрині [3].

Також зустрічаються розширення моделей, розглянутих на рис.1.1:

- СМН для несправностей, що перемежовуються (рис. 1.2а);
- АМН для несправностей, що перемежовуються (рис. 1.2б).

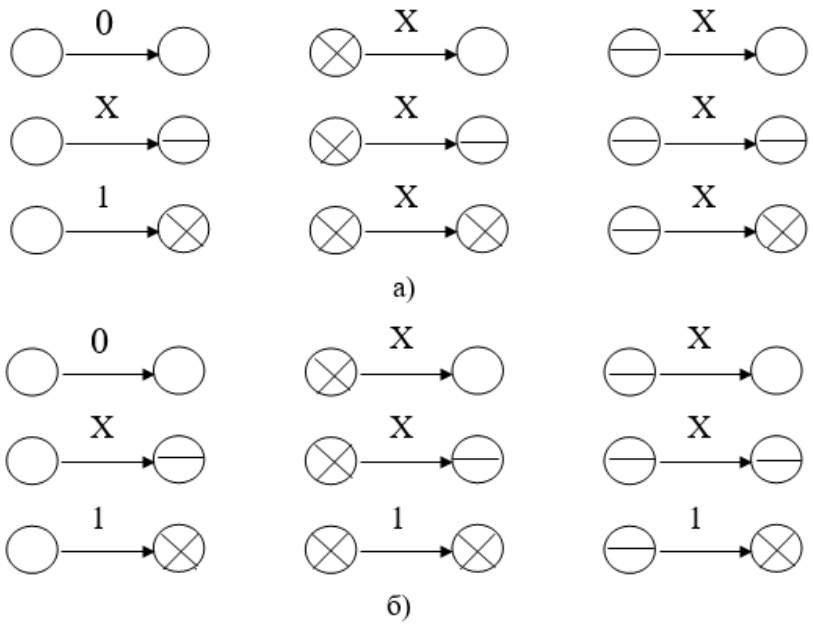


Рис.1.2. Моделі відмов, при яких несправності перемежовуються

Процес організації самодіагностування створює проблему вибору діагностичного ядра, способів його перевірки та можливості локалізації компонентів, що відмовили. Зазвичай створення діагностичного ядра здійснюється наступними способами:

1. Діагностичне ядро, що розширюється. Цей спосіб має на увазі виділення компонента системи роботоздатність якого може бути підтверджена деяким статичним ресурсом. За допомогою діагностичного ядра визначається стан усіх інших компонентів системи до останнього модуля [3].

2. Поєднання децентралізованого та централізованого керування процесами у багатомодульній системі. При централізованому управлінні значно підвищуються вимоги до надійності керуючого вузла, проте такий спосіб призводить до спрощення програмного забезпечення всієї системи. Для децентралізованого керування процесами, навпаки, зростають вимоги до програмного забезпечення, оскільки кожний з компонентів може виконувати керуючі функції та проводити діагностування інших компонентів [3].

При створенні багатопроцесорної системи необхідно враховувати її характеристики та забезпечити оптимальний рівень децентралізації всієї системи.

Самодіагностування може бути виконане наступним чином:

1. Паралельний пошук елементів системи, що відмовили. Самодіагностування у багатопроцесорній системі можна вважати паралельним, якщо всі модулі системи можуть бути визначені без заміни несправних на справні.

2. Послідовний пошук елементів, що відмовили. Самодіагностування у багатопроцесорній системі можна вважати послідовним, якщо хоча б один модуль, що відмовив, можна визначити без заміни на справний [3].

1.3. Висновки

У даному розділі було обґрунтовано актуальність обраної теми дослідження. Розглянуто питання проектування багатопроцесорних систем та забезпечення їх надійності. Пояснена необхідність забезпечення відмовостійкості у багатопроцесорних системах, а також розглянута задача самодіагностування.

Розглянуто способи організації самодіагностування у багатопроцесорних системах (паралельне або послідовне), а також надано загальну характеристику існуючих моделей несправностей.

2. МЕТОД ЗМЕНШЕННЯ ЧАСУ САМОДІАГНОСТУВАННЯ У БАГАТОПРОЦЕСОРНИХ СИСТЕМАХ

2.1. Вибір моделі несправностей

У попередньому розділі було розглянуто існуючі моделі несправностей та їх області застосування. Для розробки методу самодіагностування необхідно обрати модель несправностей. Розглянемо модель Препарата-Метца-Чена.

При діагностуванні великої системи на наявність відмов, вона, як правило, розбивається на компоненти або модулі. Вони не повинні бути ідентичними, хоча мають бути достатньо потужними для перевірки іншого модуля. Після процесу тестування модуль, що тестувався, отримує статус справного або несправного. Звичайно, цей статус має сенс тоді, коли тестуючий модуль є справним. В іншому випадку, результат тесту сприймається як ненадійний.

Якщо провести тестування системи в деякій послідовності та об'єднати результати тестування, вони можуть бути зображені як орієнтований граф з бінарною вагою. Тоді кожен модуль u_i системи – вершина графа, а зв’язок b_{ij} відповідає тесту, в якому u_i тестує u_j. A_{ij}=1, якщо u_i справний; A_{ij}=0, якщо несправний. Якщо u_i несправний, результат тесту є ненадійним, припускається, що a_{ij} приймає значення {0, 1} і не залежить від стану u_j [4].

Визначення 1. Множина з’єднань b_{ij} ($i, j, = 1, 2, \ldots, n$) є зв’язками системи, і записується, як матриця зв’язків $C \equiv \|c_{ij}\|$, яка визначається

$$c_{ij} = \begin{cases} 1 & \text{,якщо зв’язок bij існує} \\ 0 & \text{,якщо bij не існує.} \end{cases}$$

Визначення 2. Стан системи характеризується множиною результатів випробувань a_{ij}; a_{ij} може бути визначена, якщо відповідний зв’язок b_{ij} існує [4].

Приклад. Розглядається система, яка складається з 5-и компонентів u_1, u_2, …, u_5 зі з'єднаннями b_{12}, b_{23}, b_{34}, b_{45} та b_{51} (рис.2.1). Матриця з'єднань C для системи наступна:

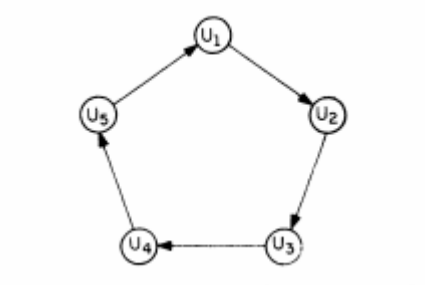


Рисунок 2.1 – Система, що складається з 5 модулів.

Стан системи задається 5-бітовим вектором, де $(a_{12}, a_{23}, a_{34}, a_{45}, a_{51})$, порядок чисел у індексі відповідає напрямку дуги, утвореної зв'язками системи.

Нехай один з модулів, u_1, несправний. Тоді

$$a_{23} = a_{34} = a_{45} = 0$$

$$a_{51} = 1$$

Тобто модуль u_5 вірно визначає стан u_1 (несправний), отже,

$$a_{12} = x, \text{тобто} \quad 0 \text{ або } 1$$

Оскільки u_1 несправний, стан модуля u_2 невідомий.

Таким чином стан для одного модуля, що відмовив, подається у наступному вигляді

$$x \ 0 \ 0 \ 0 \ 1$$

або одним із варіантів циклічних перестановок. Якщо за послідовністю нулів на позиціях …, $a_{k-3,\,k-2}$, $a_{k-2,\,k-1}$ є 1 на позиції $a_{k-1,k}$, цей результат тесту вважається надійним, і визначає модуль u_k несправним. Отже, послідовність, що має вигляд 0 0 0 1 «вказує» на модуль, що відмовив, оскільки дуга, вага якої 1, правильно визначає несправний модуль.

Продемонстровано, що система може діагностувати будь-яку одиничну відмову. Якщо розглядати питання визначення більше одного несправного модуля, то можливі 2 ситуації:

1. Здатність виявлення двох несправностей одночасно.

2. Здатність визначення щонайменше одного модуля, що відмовиви, при можливих двох.

Визначення 3. *T*-відмовостійка система з *n* модулів може виявити несправний модуль за один цикл роботи, якщо несправні модулі в системі можуть бути визначені без заміни при умові, що кількість несправних модулів не перевищує значення *t* [4].

Визначення 4. Система з *n* модулів називається послідовно-діагностованою з визначенням *t*-несправностей, якщо хоча б один модуль, що відмовив, може бути визначений без заміни при умові, що кількість несправних модулів не перевищує значення *t*.

Кожна система, яка може визначити *t*-несправностей за один цикл роботи, також вважається послідовною *t*-стійкою до відмов системою. Для демонстрування існування системи, яка може вважатися послідовно діагностованою з визначенням *t*-несправностей, але при цьому не виявляє несправність за один цикл, необхідно повернутися до рис.2.1. Нехай u_1 та u_2 несправні. Множина тестових зв'язків наведена на рис.2.2(а). Множина зв'язків для несправного модуля u_1, наведена на рис.2.2(б). Ці ситуації не можуть бути диференційовані. Отже, система, наведена на рис.2.1., не може виявити 2 несправності за один цикл роботи [4].

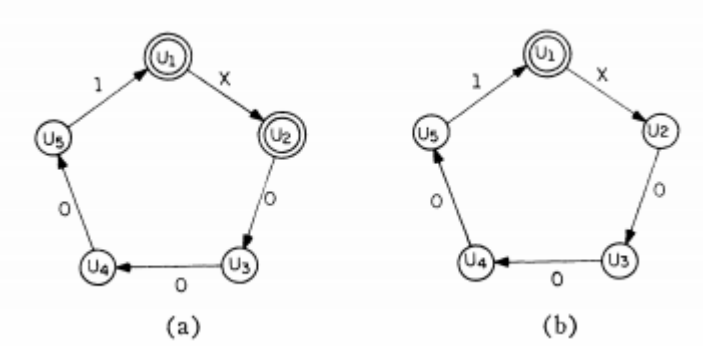


Рисунок 2.2 – Стан системи, описаний двома різними конфігураціями. Несправні модулі позначені подвійним кружечком.

Проте система на рис.2.2 здатна виявляти 2 несправності. Всі можливі послідовності з однією/двома відмовами зображено на рис.2.3., з точністю до циклічних перестановок, разом з результатами випробувань. Якщо стан має послідовність нулів та одиниць - 001, то процесор, на який вказує така послідовність, несправний. У цьому випадку послідовності x1101 модуль, на який вказує ця послідовність 1101, є несправним. Отже, несправний модуль може бути ідентифікованим у будь-якому випадку. Подальші тести необхідні для визначення несправності іншого модуля та його оцінки.

Несправні модулі	Жодного	u1	u1,u2	u1, u3
Стан	OOOO	XOOO1	XXOO1	X1XO1

Рисунок 2.3. – Всі одиничні та подвійні послідовності системи на Рис. 1.

2.1.1. Системи, що здатні визначити t-несправностей за один цикл роботи

Діагностована система *S* визначається матрицею з’єднань *C(n×n)*, порядок яких характеризує кількість модулів, на які може розкладатися система *S*. Досліджується зв'язок між *n* і *t* та числом несправних модулів для системи, яка може виявити несправність за один цикл роботи.

Теорема 1. Припускається, що система *S* може діагностувати *t*-несправностей за один цикл роботи. Тоді $n \geq 2t+1$. Якщо $n \geq 2t+1$, завжди є

спосіб сформувати таку систему *S*, яка зможе діагностувати *t*-несправностей за один цикл роботи [4].

Доведення. Будується максимально зв'язний двонаправлений граф, тобто, зв'язуються усі пари по *n* модулів. Однією з характеристик зв’язного графа є те можливість створення циклу, який сполучає будь-яку підмножину модулів. Для будь-якого циклу *Z* модулів з усіма тестовими зв’язками зі значенням 0, z-компоненти в цьому циклі або всі справні, або навпаки. Якщо $z \geq t+l$, всі процесори в циклі мають бути справними, інакше порушується умова максимальної кількості несправних модулів. Якщо в петлі знаходиться $t+1$ або більше справних модулів, то процес діагностики завершується, так як будь-які визначені справні модулі можуть визначити знаходження всіх модулів, що відмовили через зв’язки. Отже, система повинна мати хоча б $t+1$ справних модулів, якщо система *S* має не більше *t* модулів, що відмовили; Отже, існування циклу з $t+1$ або більше справних модулів підтверджується.

Для системи *S* з $n < 2t+1$ процесорами та довільними тестовими зв'язками, існування двох різних моделей несправностей може призвести до тієї ж конфігурації і не завжди система *S* здатна відрізнити наведені послідовності несправностей, тому система *S* вважається не придатною до діагностування *t*-несправностей за один цикл роботи програми.

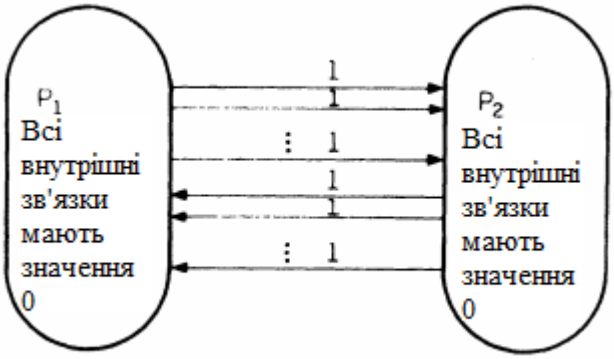


Рисунок 2.4 – Недіагностована ситуація

Теорема 2. В системі *S*, що здатна діагностувати *t*-несправностей за один цикл роботи, модуль, що перевіряється, має тестуватися щонайменше *t* іншими модулями [4].

Визначення 5. Система *S*, що здатна визначити *t*-несправностей за один цикл роботи, вважається оптимальною, якщо кожний модуль тестується рівно *t* іншими модулями, а $n=2t+1$.

Визначення 6. Система *S* належить до $D_{\delta t}$, якщо існує дуга u_i з u_j, $j-i=\delta m \pmod n$, де m - *1, 2, …, t.*

На рис.2.5 показано 2 типи системи для $n=5$. D_{12} зображена на рис.2.5(а), D_{22} – на рис.2.5(б). Система може визначити *t*-несправностей за один цикл при використанні конфігурації $D_{\delta t}$, де $(\delta, n)=1$, δ та n взаємно прості.

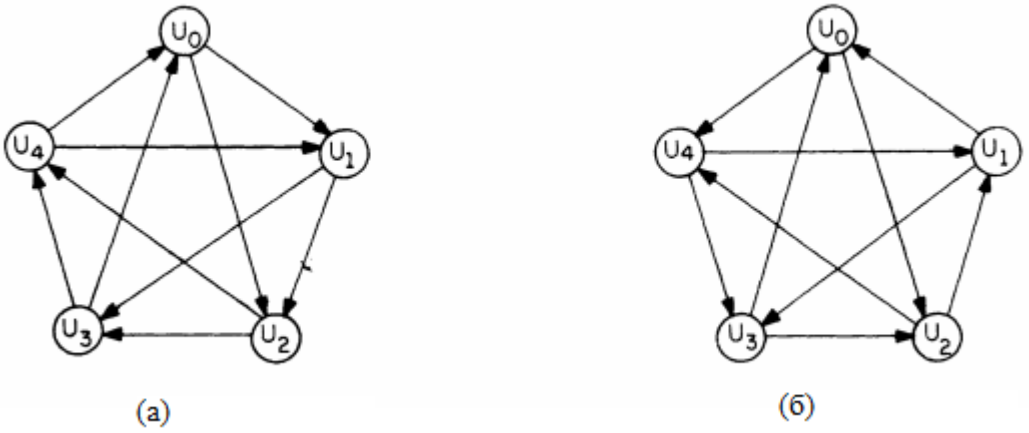


Рисунок 2.5 – Два типи конфігурацій для діагностування *t*-несправностей за один цикл роботи

Теорема 3. При використанні системою S конфігурації $D_{\delta t}$, де $(\delta, n)=1$, система може визначити *t*-несправностей за один цикл, а $D_{\delta t}$ вважається оптимальною конфігурацією.

2.1.3. Послідовно діагностовані системи

З попереднього пункту відомо, що навіть для оптимального графа, необхідно nt зв’язків виявити несправність за один цикл роботи. Інтерес до

послідовно-діагностованих систем пояснюється тим, що такі системи потребують менше зв’язків для тестування. Теорема 1 справедлива і для цього типу систем.

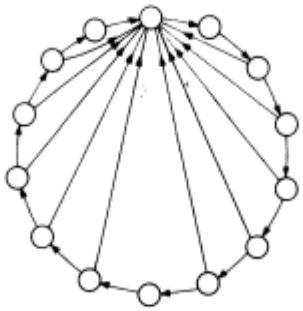


Рисунок 2.6 – Приклад послідовного з’єднання для $t = 6$ і $n = 14$

Теорема 4. Існує такий клас графів з $N = N+ 2t-2$, що можуть визначити t-несправностей при послідовному діагностуванні [4].

Приклад. На рис.2.6 зображене послідовне з'єднання, відповідно до вимог Теореми 4 при t=6 та $n=14$.

2.2. Аналіз існуючих методів організації самодіагностування у багатопроцесорних системах

Проведемо аналіз існуючих методів самодіагностування, які використовують модель Препарата-Метца-Чена, як модель несправностей.

2.2.1. Метод діагностування модулей, що відмовили у цифрових системах за допомогою ланцюжків з трьох модулей

Методи системного діагностування відмов у багатопроцесорних системах ґрунтуються на наступному: при зменшенні максимальної кількості процесорів, що можуть одночасно відмовити, необхідні функції діагностування реалізуються за допомогою справних машин. Для опису взаємодії окремих частин системи при реалізації діагностичних процедур розроблені різні моделі несправностей[1]. Даний метод використовує модель Препарата-Метца-Чена. Задача дешифрування реального синдрому R полягає

в наступному: на основі його значення і відомої структури тестових зв'язків системи визначаються модулі, які відмовили. Відомі два основних способи організації діагностування: однократний, коли все відмовили модулі достовірно діагностуються за результатами виконання тестів, і послідовний (або багаторазовий з заміною). [5] Нижче пропонується варіант однократного діагностування модулів, що відмовили, в цифрових системах, заснований на використанні діагностичних властивостей ланцюжків з трьох модулів (рис.2.7).

Для отримання шуканих кон'юнкцій, необхідних для дешифрування реального синдрому R, використовуються основні положення аналітичного методу. Sk (двійкова змінна) позначається технічний стан модуля mk і приймемо, що Sk = 1, якщо mk відмовив, і Sk = 0 (Sk = 1), якщо mk справний [5].

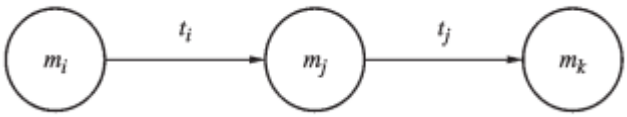


Рисунок 2.7 – Ланцюжок з трьох модулів

Звідси, на основі результату rj тесту tj, за допомогою якого mj перевіряє mk (Рис. 1), технічний стан цих модулів описується наступною функцією:

$F_j = S_k r_j \vee S_k r_j \vee S_j$. (1)

Для визначення технічного стану системи, що складається з n модулів, за результатами $r1, r2, ..., rl$ тестів $t1, t2, ..., tl$ необхідно побудувати таку логічну функцію Ф ($S1, S2, ..., Sn$), яка буде кон'юнкцією всіх функцій Fj:

Ф ($S1, S2, ..., Sn$) =

$l \cap j = 1\ Fj$. (2)

Виконавши над Ф ($S1, S2, ..., Sn$) з (2) перетворення ΠΣ → ΣΠ [1], відбувається перехід від кон'юнктивної нормальної форми (КНФ) до диз'юнктивної нормальної форми (ДНФ). Обравши частину ДНФ з найменшим

числом змінних $S1, S2, ..., Sn$ в прямій формі, тобто без заперечень, знаходиться найменша сукупність модулів, що могли відмовити. Логічна функція (1) до пари під функцій перетворюється наступним чином:

$Fj = Sk\ rj \vee Skrj \vee Sj\ (rj \vee rj) = fj0 \vee fj1$, (3), де

$fj0 = rj\ (Sk \vee Sj)$, $fj1 = rj\ (Sk \vee Sj)$. (4)

На базі під функцій $fj0$, $fj1$ з (4) будуються наступні функції:

$rj \rightarrow Sk \vee Sj, rj \rightarrow Sk \vee Sj$, (5)

які мають наступні значення: якщо результатом тесту tj є rj, то модуль mk - справний або відмовив модуль mj; якщо ж результатом тесту tj є rj, то відмовив модуль mj, або mk, або обидва. З (5) випливає, що за результатом одного тесту формується висновок про стан двох модулів, проте точності діагнозу не вистачає [5]. Тому необхідно використовувати систему тестів, в якій є хоча б один тест для перевірки кожного модуля системи, що аналізується. При цьому для отримання загального діагнозу до множини наборів функцій виду (5), сформованих для результатів виконання тестів $r1$, $r2$, ..., rl, необхідно виконати перетворення, які визначаються формулою (2). Замість цього розглядається ланцюжок з трьох модулів і двох тестів (ti, tj) (рис. 1), де контролюючий модуль mi тестує модуль mj за допомогою ti, а контролюючий модуль mj перевіряє модуль mk за допомогою tj. Цим тестам (ti, tj) відповідають дві функції Fi і Fj виду (3), (4) і чотири наступні функції виду (5):

$ri \rightarrow Sj \vee Si; rj \rightarrow Sk \vee Sj;$

$ri \rightarrow Sj \vee Si; rj \rightarrow Sk \vee Sj$.(6)

Таблиця 1.1 – Результати тестування

№№ п/п	m_i	m_j	m_k	r_i	r_j	Диагноз	Оценка
0	Испр	Испр	Испр	0	0	m_k – испр	правильно
1	Испр	Испр	Отказ	0	1	m_k – отказ	правильно
2	Испр	Отказ	Испр	1	X	ϕ	–
3	Испр	Отказ	Отказ	1	X	ϕ	–
4	Отказ	Испр	Испр	$X(=0)$	0	m_k – испр	правильно
5	Отказ	Испр	Отказ	$X(=0)$	1	m_k – отказ	правильно
6	Отказ	Отказ	Испр	$X(=0)$	$X(=0)$	m_k – испр	правильно
7	Отказ	Отказ	Испр	$X(=0)$	$X(=1)$	m_k – отказ	**ошибка**
8	Отказ	Отказ	Отказ	$X(=0)$	$X(=0)$	m_k – испр	**ошибка**
9	Отказ	Отказ	Отказ	$X(=0)$	$X(=1)$	m_k – отказ	правильно

На підставі функцій (6) утворюємо наступні кон'юнкції:

$rirj \rightarrow (Sj \vee Si)\ (Sk \vee Sj) = SjSk \vee SiSk \vee SiSj;$

$rirj \rightarrow (Sj \vee Si)\ (Sk \vee Sj) = SjSk \vee SiSk \vee SiSj.$ (7)

Права частина кожної з формул містить по три кон'юнкції; це означає, що відповідні технічні стани модулів, що входять до кожної кон'юнкцію, можуть призвести до одного і того ж результату тестування, що наданий в лівій частині формул (7). Для отримання достовірних результатів необхідно, аби контролюючі модулі mi і mj були справні; тоді формули (7) спрощуються і зводяться до наступних кон'юнкцій:

$rirj \rightarrow Sk, rirj \rightarrow Sk,$ (8)

якщо обидва тести (ti і tj) дали нульовий результат ($rirj$ = 1), то модуль mk справний; якщо ж тест tj дав одиничний результат при 0-результаті тесту ti ($rirj$ = 1), то модуль mk відмовив [5].

Детальніше розглядаються можливі результати діагностування модуля mk за допомогою кон'юнкцій виду (8) при різних технічних станах контролюючих модулів mi і mj. У перших трьох стовпцях таблиці наведені

можливі стану модулів mi, mj і mk; в двох наступних - можливі результати тестування модулів mj і mk відповідно; далі надано результат діагностування технічного стану модуля mk з використанням кон'юнкцій (8) і його оцінка (шляхом порівняння зі стовпцем mk). У рядках 4-9 показані не всі можливі заміни X на 0 і 1, а тільки такі, при яких можливо виконання кон'юнкцій виду (8). Як видно з рядків 2 і 3, середнє арифметичне значення ri перешкоджає отриманню будь-якого діагнозу. Слід зазначити, що результати діагнозу, показання в рядках 4-9, свідомо недостовірні, так як отримані з використанням модулів, що відмовили, в якості контролюючих. Проте до діагностичного експерименту, множина модулей, що відмовили, невідома, і завдання дешифрування реального синдрому в тому і полягає, щоб визначити модулі, що відмовили, на основі достовірної частини синдрому [5].

За даними таблиці 1.1 можна зробити наступні висновки:

- кон'юнкції виду (8) забезпечують однозначне діагностування технічного стану модуля mk в ланцюжку, що включає справні модулі mi і mj (рядки 0 і 1);

- наявність в ланцюжку одного модуля, що відмовив, (mi або mj) може перешкоджати діагнозу (символ φ в рядках 2 і 3 означає, що діагноз відсутній) або, навпаки, призвести до правильного діагнозу технічного стану модуля mk (рядки 4 і 5);

- наявність в ланцюжку двох модулів, що відмовили (і mi, і mj) може дати як правильний (рядки 6 і 9), так і неправильний діагноз технічного стану модуля mk (рядки 7 і 8).

Як відомо [2], структура p-систем, що даігностуються один раз, (p-ОДС) повинна відповідати наступним вимогам:

- число модулів $n \geq 2p + 1$,

- кожен модуль перевіряється не менше, ніж p тестами.

З цих вимог випливає, що для кожного модуля можна збудувати не менше, ніж $p2$ ланцюжків з трьох модулів, подібних до рис. 1.1. Тоді загальне число (N) вихідних виразів виду (8) для такої системи дорівнює: $N = np2$.

Використання всіх N ланцюжків забезпечує одноразове діагностування всіх s модулів, що відмовили, у системі за умови, що їх кількість не перевищує величину p, яку називають мірою діагностування цифрової системи [5].

Кількість використовуваних ланцюжків з трьох модулів скорочується за допомогою особливостей діагностичних графів p-ОДС з регулярною структурою. Для кожного модуля p-ОДС, що включає $n(= 2p + 1)$ модулів, будуються p ланцюжків з трьох модулів так, що вони не мають інших спільних модулів, крім k-го, для перевірки якого вони побудовані. Для цього використовується ланцюжки настпуного вигляду:

$mk\text{-}2p \rightarrow mk\text{-}p \rightarrow mk,$

.....

$mk\text{-}p\text{-}l \rightarrow mk\text{-}l \rightarrow mk,$

(9) ... .

$mk\text{-}p\text{-}1 \rightarrow mk\text{-}1 \rightarrow mk,$

де k = 1,2, ..., $2p + 1 = n$, а віднімання номерів в індексах проводиться за mod ($2p$ + 1).

В якості базової ситуації відмов розглядається Nx, що містить s (= p) модулів, що відмовили. Ці s модулів розподілені по p ланцюжках виду (9), побудованим для k-го модуля (k = 1, ..., $2p + 1 = n$) [5].

Ланцюжки, що містять два справних контрольних модуля, мають назву ланцюжками 0-типу, а їх кількість в p ланцюжках для k-го модуля позначається величиною $ck0$. Аналогічно, ланцюжки, що містять один або два модуля, що відмовили в тих самих p ланцюжках для k-го модуля – ланцюжки

1-го типу (2-го типу), а їх кількість позначається відповідно $ck1$ і $ck2$. Тоді для ланцюжків виду (9) справедлива рівність:

$ck0 + ck1 + ck2 = p$. (10)

Вводиться два параметри: α = 1,2, ..., 1 2 Показати (p - 1), β = 0,1, ..., α, і наступне співвідношення:

$s = 2\alpha + 1$. (11)

Розглядаються окремо два випадки: k-й модуль відмовив, і k-й модуль справний.

У першому випадку серед p ланцюжків k-го модуля міститься: s-1 = 2α контролюючих модуля, що відмовили. Нехай $ck2 = \alpha - \beta$, тоді $ck1 = 2\alpha - 2ck2 = 2\beta$, та на підставі (10 і 11):

$ck0 = p\ (= s) - ck2 - ck1 = 2\alpha + 1 - (\alpha - \beta) - 2\beta = \alpha - \beta + 1 = ck2 + 1$. (12)

У другому випадку серед p ланцюжків k-го модуля міститься: $s = 2\alpha + 1$ контролюючих модуля, що відмовили. Нехай $ck2 = \alpha - \beta$, тоді $ck1 = 2\beta + 1$, і на підставі (10 і 11):

$ck0 = p\ (= s) - ck2 - ck1 = 2\alpha + 1 - (\alpha - \beta) - (2\beta + 1) = \alpha - \beta = ck2$. (13)

Співвідношення (12) показує, що при відмові k-о модуля, $ck2$ ланцюжки 2го типу можуть дати помилковий діагноз (назвати k-й модуль справним), але $ck0$ (= $ck2$ + 1) ланцюжків 0-типу, що містять справні контролюючі модулі, дадуть більшу кількість правильних діагнозів (назвуть k-й модуль несправним). У свою чергу, співвідношення (13) показує, що при справному k-у модулі $ck2$ ланцюжки 2-го типу можуть дати помилковий діагноз (назвати k-й модуль несправним), і $ck0$ (= $ck2$) ланцюжків 0-типу, що містять справні контролюючі модулі, дадуть однакову кількість правильних діагнозів (назвуть k-й модуль справним).

Розглядаються зміни в співвідношеннях (12, 13) при $s < p$, тобто при $p = s + \gamma$, де $\gamma \geq 1$. Умова ($s < p$) не змінює оцінок $ck2$ і $ck1$, але змінює $ck0$.

Так, для випадку, при якому k-й модуль відмовив, отримаємо замість (12):

$c1\ k0 = s + \gamma - ck1 - ck2 = 2\alpha + 1 + \gamma - (\alpha - \beta) - 2\beta = \alpha - \beta + 1 + \gamma$. (12а)

Якщо ж k-й модуль справний, то отримаємо замість (13):

$c0\ k0 = s + \gamma - ck1 - ck1 = 2\alpha + 1 + \gamma - (\alpha - \beta) - (2\beta + 1) = \alpha - \beta + \gamma$. (13а)

Порівняння отриманих оцінок з $ck0$ з (12), (13) демонструє, що при зменшенні потужності ситуацій з відмовами на величину γ, збільшується кількість ланцюжків 0-типу на ту ж величину γ. Розглядається інший варіант системи, в якому $n > 2p + 1$, тобто

$n = 2p + 1 + m$, де $m \geq 1$.

Це значить, що в ланцюжках виду (9), побудованих для k-го модуля, не будуть залучені m модулів. Нехай серед цих m модулів міститься γ ($0 \leq \gamma \leq m$) модулей, що відмовили. Тоді при $\gamma = 0$, тобто всі m модулів справні, справедливі співвідношення (12), (13) і висновки, сформульовані для базового варіанту. Якщо ж $\gamma \geq 1$, то для діагностування інших ($p - \gamma$) модулів, що відмовили, та які входять в ланцюжки виду (9), справедливі співвідношення (12а), (13а) і висновки, отримані для таких ситуацій [5].

Розглянуті варіанти дозволяють зробити висновок, що базовий варіант ($n = 2p + 1$, а $s = p$) є так званою нижньою границею цього методу однократного діагностування модулів, що відмовили в p-ОДС. Відхилення загального числа модулів ($n > 2p + 1$) або числа модулів, що відмовили ($s < p$) збільшує число ланцюжків 0-типу а, отже, і кількість достовірних результатів діагностування модулів, що відмовили. В результаті зростає достовірність процедури дешифрування реального синдрому. Наведені вище співвідношення (12), (12а), (13), (13а) дозволяють обґрунтувати наступну процедуру дешифрування реального синдрому, отриманого при перевірці p-ОДС за допомогою системи тестів, що реалізують ланцюжки виду (9) для перевірки кожного модуля:

1) виконується первинний діагноз: для отриманого реального синдрому $R0$ за допомогою системи кон'юнкцій виду (8) обчислюється результат діагностування стану кожного модуля системи;

2) формується підмножина модулів, що можуть відмовити, на підставі наступного правила:

- для k-го модуля (k = 1, ..., n) розраховуються числа $dk1$ позитивних (Sk=1) і $dk0$ негативних (Sk=1) висновків про його технічний стан. Якщо різниця ($dk1$ - $dk0$) ≥ 1, то k-й модуль включається до складу підмножини модулів, що могли відмовити;

3) корегується вихідний синдром $R0$. Виключаються значення реакцій ri і rj, які отримані з використанням модулів, що можуть відмовити, в якості контролюючих;

4) виконується вторинний діагноз: для скоригованого реального синдрому $R1$ 0 за допомогою системи кон'юнкцій виду (8) обчислюється стан кожного модуля системи. Запропонована процедура дешифрування скінченна, та не містить більше, ніж два цикли дешифрування реального синдрому, і закінчується отриманням достовірних і несуперечливих результатів [5].

Розглядається застосування запропонованого методу дешифрування синдрому на прикладі системи 3-ОДС (p = 3, n = 7). На рис.2.8 представлений діагностичний граф системи, де вершини позначені номерами модулів: 1, 2, ..., 7, а дуги визначають пари модулів (i, j), (j, k), що беруть участь у виконанні тестів ti, j , tj, k відповідно, де перший номер пари i (j) визначає контролюючий модуль, а другий номер j (k) – модуль, що тестується. На рис.2.8 для прикладу деякі дуги позначені індексами тих тестів, яким вони відповідають: 1,2; 3,4; 3,6; 5,6. Подібні позначення модулів і тестів будуть використовуватися і надалі, щоб спростити структуру позначень, тобто номер i замість модуля mi, індекс i, j замість тесту ti, j.

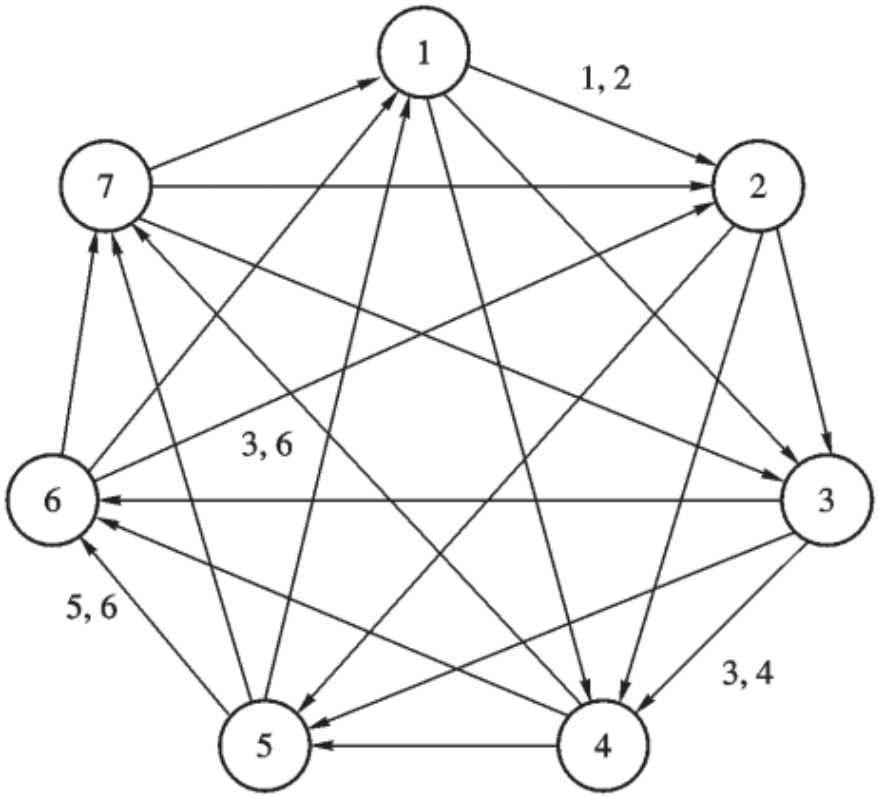


Рис.2.8 – Діагностуючий граф

Ланцюжки виду (9), побудовані для 3-ОДС і записані з урахуванням введених позначень, наведені нижче:

$$
\begin{array}{llll}
3 \to 6 \to 7; & 2 \to 5 \to 6; & 1 \to 4 \to 5; & 7 \to 3 \to 4; \\
2 \to 5 \to 7; & 1 \to 4 \to 6; & 7 \to 3 \to 5; & 6 \to 2 \to 4; \\
1 \to 4 \to 7; & 7 \to 3 \to 6; & 6 \to 2 \to 5; & 5 \to 1 \to 4; \\
6 \to 2 \to 3; & 5 \to 1 \to 2; & 4 \to 7 \to 1; & \\
5 \to 1 \to 3; & 4 \to 7 \to 2; & 3 \to 6 \to 1; & \\
4 \to 7 \to 3; & 3 \to 6 \to 7; & 2 \to 5 \to 1. &
\end{array} \qquad (14)
$$

Як зазначалося раніше, кожному ланцюжка типу (9) відповідає пара кон'юнкцій виду (8). При написанні системи кон'юнкцій (15) для ланцюжків (14) використані наступні спрощені правила запису реакцій ri, j, rj, k і станів Sk, Sk:

$\overline{i,j}\,\overline{j,k}$ $(\overline{i,j}\,j,k)$ вместо $\overline{r}_{i,j}\overline{r}_{j,k}\,(\overline{r}_{i,j}r_{j,k})$
и $\overline{k}(k)$ вместо $\overline{S}_k(S_k)$.

Система кон'юнкцій (15):

1.	$\overline{3,6}\,\overline{6,7}\to\overline{7};$	13.	$\overline{1,4}\,\overline{4,5}\to\overline{5};$	25.	$\overline{6,2}\,\overline{2,3}\to\overline{3};$	37.	$\overline{4,7}\,\overline{7,1}\to\overline{1};$
2.	$\overline{3,6}\,6,7\to 7;$	14.	$\overline{1,4}\,4,5\to 5;$	26.	$\overline{6,2}\,2,3\to 3;$	38.	$\overline{4,7}\,7,1\to 1;$
3.	$\overline{2,5}\,\overline{5,7}\to\overline{7};$	15.	$\overline{7,3}\,\overline{3,5}\to\overline{5};$	27.	$\overline{5,1}\,\overline{1,3}\to\overline{3};$	39.	$\overline{3,6}\,\overline{6,1}\to\overline{1};$
4.	$\overline{2,5}\,5,7\to 7;$	16.	$\overline{7,3}\,3,5\to 5;$	28.	$\overline{5,1}\,1,3\to 3;$	40.	$\overline{3,6}\,6,1\to 1;$
5.	$\overline{1,4}\,\overline{4,7}\to\overline{7};$	17.	$\overline{6,2}\,\overline{2,5}\to\overline{5};$	29.	$\overline{4,7}\,\overline{7,3}\to\overline{3};$	41.	$\overline{2,5}\,\overline{5,1}\to\overline{1};$
6.	$\overline{1,4}\,4,7\to 7;$	18.	$\overline{6,2}\,2,5\to 5;$	30.	$\overline{4,7}\,7,3\to 3;$	42.	$\overline{2,5}\,5,1\to 1.$
7.	$\overline{2,5}\,\overline{5,6}\to\overline{6};$	19.	$\overline{7,3}\,\overline{3,4}\to\overline{4};$	31.	$\overline{5,1}\,\overline{1,2}\to\overline{2};$		
8.	$\overline{2,5}\,5,6\to 6;$	20.	$\overline{7,3}\,3,4\to 4;$	32.	$\overline{5,1}\,1,2\to 2;$		
9.	$\overline{1,4}\,\overline{4,6}\to\overline{6};$	21.	$\overline{6,2}\,\overline{2,4}\to\overline{4};$	33.	$\overline{4,7}\,\overline{7,2}\to\overline{2};$		
10.	$\overline{1,4}\,4,6\to 6;$	22.	$\overline{6,2}\,2,4\to 4;$	34.	$\overline{4,7}\,7,2\to 2;$		
11.	$\overline{7,3}\,\overline{3,6}\to\overline{6};$	23.	$\overline{5,1}\,\overline{1,4}\to\overline{4};$	35.	$\overline{3,6}\,\overline{6,2}\to\overline{2};$		
12.	$\overline{7,3}\,3,6\to 6;$	24.	$\overline{5,1}\,1,4\to 4;$	36.	$\overline{3,6}\,6,2\to 2;$		

Порівняння записів з (15) із ланцюжками (14) демонструє легкість переходу від ланцюжків виду (9) до кон'юнкцій виду (8). Приймемо наступний порядок запису реакцій ri, j в векторі потенційного синдрому:

Тоді вектор Rx матиме наступний вигляд:

$$1,4_12,5_23,6_34,7_45,1_56,2_67,3_71,3_82,4_93,5_{10}4,6_{11}5,7_{12}6,1_{13}7,2_{14}1,2_{15}2,3_{16}3,$$
$$4_{17}4,5_{18}5,6_{19}6,7_{20}7,1_{21},$$

де індекс кожної пари чисел вказує порядковий номер цієї пари в прийнятому розташуванні.

Припускається, що в системі існує відмова $N1$=($m1$,$m4$,$m5$). Відповідно до моделі Препарата-Метца-Чена цієї ситуації відповідає потенційний синдром
$Rx1$=X11203X4X50607X819110X11X12113014X15016117X18X19020121, а одним із способів реалізації якого, при проведенні діагностичних перевірок, може бути реальний синдром $R11$ наступного виду:

$$R_{11} = 0_1 1_2 0_3 1_4 0_5 0_6 0_7 1_8 1_9 1_{10} 1_{11} 0_{12} 1_{13} 0_{14} 1_{15} 0_{16} 1_{17} 0_{18} 0_{19} 0_{20} 1_{21}.$$

Для наведених значень $r1$, ..., $r21$ вектора $R11$ відповідно до пункту 1 запропонованої процедури одержано результати первинного діагнозу, обчислені за допомогою системи кон'юнкцій (15) технічних стану модулів 1, ..., 7. З 42 кон'юнкцій системи (15) спрацюють і дадуть діагностичні висновки 15 наступних (далі названі номери кон'юнкцій та її діагноз): 1 (7), 6 (7), 10 (6), 11 (6), 13 (5), 16 (5), 18 (5), 20 (4), 22 (4), 23 (4), 25 (3), 28 (3), 32 (2), 35 (2), 40 (1). Відповідно до п. 2 розраховується різниця ($dk1$ - $dk0$) і формується підмножина модулів, що можуть відмовити, включивши до неї ті модулі, для яких ця різниця більша нуля. В даному випадку модулі, що підозрюються у відмові, - 5, 4, 1 [5].

Згідно з п. 3 викреслюється з синдрому $R11$ ті реакції ri, j, де індекс i приймає значення 1, 4, 5, тобто пари номерів: 1,41, 1,38, 1,215, 4,74, 4,611, 4,518, 5,15, 5,712, 5,619. Проте таке викреслювання не додає нових значень у вторинний діагноз в порівнянні з первинним. В результаті залишилися достовірні значення реакцій справних модулів 2, 3, 6, 7, що можуть забезпечати вірний вторинний діагноз: 1 (7), 11 (6), 16 (5), 18 (5), 20 (4), 22 (4), 25 (3), 35 (2), 40 (1), тобто вони можуть підтвердити раніше отримані висновки про підмножини модулів, які відмовили, (1, 4, 5) і справність інших модулів (7, 6, 3, 2).

Тепер розглядається інша відмова $N2$ = ($m1$, $m3$, $m5$), для якої однієї з реалізацій відповідного потенційного синдрому $Rx2$ може бути реальний синдром $R22$ наступного вигляду:

$R22$ = 01120304050617080911001101211301411511601711801902012 1.

Діючи у тому ж порядку, обчислюється значення первинного діагнозу: 1 (7), 9 (6), 14 (5), 18 (5), 21 (4), 23 (4), 26 (3), 27 (3) , 30 (3), 32 (2), 33 (2), 35 (2), 38 (1), 40 (1); підрахувавши для цих записів різницю ($dk1$ -$dk0$), формується підмножина модулів, що могли відмовити: 5, 3, 1. Далі з синдрому $R22$

викреслюються реакції 1,41, 1,38, 1,215, 3,63, 3,510, 3,417, 5 , 15, 5,712, 5,619 і, повторивши обчислення кон'юнкцій, записується вторинний діагноз: 18 (5), 21 (4), 26 (3), 30 (3), 33 (2), 38 (1). Як видно з цих записів, другий цикл підтвердив висновок про відмову модулів (5, 3, 1) і справності двох модулів (2, 4). Що стосується модулів 6 і 7, то модулі, що відмовили, у ситуації $N2$ розташувалися таким чином, що всі три ланцюжки, побудовані для модулів 6 і 7, виявилися 1-го типу та відповідні їм реакції були викреслені при проведенні корекції. Розглядається ще одна ситуація $N3 = (m3, m6)$, яка відповідає варіанту ($s = 2$) <($p = 3$). Одна з реалізацій відповідного потенційного синдрому може мати вигляд:

$R33$ = 010203040516171809110111012113014015116117018119120021.

Після обчислення значень первинного діагнозу і підрахунку різниці ($dk1$ -$dk0$) в підмножині модулів, що могли відмовити, увійдуть 6 і 3, і з синдрому $R33$ викреслюються реакції: 3,63, 3,510, 3,417, 6,26, 6,113, 6,720. Обчислення кон'юнкції для такого скоригованого синдрому дає вторинний діагноз: 3 (7), 5 (7), 8 (6), 10 (6), 13 (5), 23 (4), 28 (3), 30 (3), 31 (2), 33 (2), 37 (1), 41 (1), що відрізняється від первинного трьома записами: 2 (7), 36 (2), 40 (1). Ці записи підтверджують відмову модулів 3, 6 і справний стан модулів 7, 5, 4, 2, 1.

Запропоновано спосіб однозначного діагностування s ($\leq p$) несправних модулів в p-однократно діагностованих системах, побудований на використанні системи тестів, що відповідають p ланцюжках певної структури, побудованим для перевірки кожного з $n \geq (2p+1)$ модулів системи. Розроблений метод дешифрування реального синдрому, отриманого при перевірці системи на даній системі тестів, що включає корекцію вихідного синдрому для відсіювання недостовірних реакцій, отриманих при використанні в ланцюжках контролюючих несправних модулів, і гарантує достовірність результатів вторинного діагнозу. У ряді випадків при дешифрування скоригованого

синдрому можуть виявитися невизначеними технічні стани деяких справних модулів. Наведені приклади використання запропонованої процедури дешифрування синдромів для трьох ситуацій відмов в тричі-однократно діагностованій системі [5].

2.2.2. Метод локального тестування у обчислювальних системах з циркулянтною діагностованою структурою

Метод створений на базі дослідження можливості діагностування мультипроцесорних обчислювальних систем (ОС) при відмовах кратності не більше t. Як модель несправностей використовується модель системи, запропонована в роботі[3], Препарати-Метца-Чена (ПМЧ-модель). Діагностична структура ОС представлена циркулянтним графом, який має вершинну ступінь t і число вершин N. Такий граф забезпечує визначення стану ОС при несправності, кратність якої не перевищує значення t (t-діагностована ОС), при $N \geq 2t+1$ за результатами порівняльного аналізу результатів тестування всіх модулів системи (глобальне діагностування) [6].

В роботі [1] запропоновані правила, за допомогою яких визначення технічного стану кожного модуля здійснюється за результатами тестування тільки тих модулів, які мають з ним зв’язок. Ці правила названі правилами самовизначення, а діагностика, яка базується на їх використанні - локальною. Використання локальних правил самовизначення дозволяє розробляти розподілені методи діагностування.

Показано, що для заданого t існують межі за кількістю модулів в системі, що представлена циркулянтним графом, поза якими локальні правила забезпечують самодіагностування не всіх модулів системи. Подібний стани ОС називається тупиковим, а діагностування – частковим [6].

В роботі [1] показано, що локальна t-діагностованість у випадку тупикових станів досягається введенням надмірності у кількості тестів, що

виконуються над системою. Однак тестування займає основну частину часу діагностування, так що застосування додаткових тестів зменшує час використання ОС за призначенням і на реакцію ОС на несправність. Для зменшення часу діагностування в роботі [2] запропоновано використовувати надмірність у кількості результатів тестування, що беруть участь в порівняльному аналізі з метою самодіагностування стану модулів. Для цього використовуються результати тестування, співвіднесені із структурними одиницями графа-циркулянта, так званими трикутниками тестування.

У даній роботі вивчається ефективність самовизначення модулів при використанні таких трикутників тестування.

Діагностичною моделлю обчислювальної системи, в якій допустимі множинні відмови і ненадійні тести, є орієнтований граф D = (V, E) [6]. Вершинами i ∈ V графа позначені модулі, а тестові зв'язки (i, j) ∈ E - дугами; (I, j) ∈ E, якщо модуль, який співвіднесено з вершиною i, може перевірити і оцінити стан модуля, що співвідноситься з вершиною j. Вага a (i, j) дуги (i, j) діагностичного графа відображає оцінку стану модуля j, що перевіряється, отриману перевіряючим модулем і після виконання ним відповідного тесту. Вага a (i, j) = 0, якщо модуль і вважає справним модуль j, інакше a (i, j) = 1. Відповідно до ПМЧ-моделі, результат тесту, що виконується несправним модулем, не залежить від фактичного стану модуля, що тестується.

Множина Fk, Fk ⊆ V, несправних модулів, що можуть одночасно бути присутніми в системі, називаємо образом несправностей. Для будь-якого Fk справедливо | Fk | ≤ t, де | X | позначає потужність множини X. Множину допустимих несправностей системи складають всі можливі поєднання з N модулів по n, де n набуває значення від 1 до t. Таким чином, множина {Fk} допустимих станів ОС залежить від значень t і N і позначається F (t, N).

Впорядкована сукупність оцінок, що одержується при виконанні всіх можливих тестів над системою в присутності Fk, називається синдромом, що породжується Fk, і позначається σ (Fk); σ (Fk) = {a (i, j): (i, j) ∈ E}.

Діагностична структура ОС представлена t-діагностованим циркулянтним графом (D1, t (N) -граф) з параметричних описом (N; 1, 2, ..., n, ..., t), де N = | V |, а n = 1, 2, ..., t - стрибки. У циркулянті D1, t (N) вершини пронумеровані і кожна i-я вершина (0 ≤ i ≤ N - 1) суміжна з вершинами (i ± 1) mod N, (i ± 2) mod N, ..., (i ± t) mod N, а дуга (i, (i + n)) [12] має мітку відповідного стрибка sn = n [6].

Нехай в діагностичному графі безлічі X (i), Y (i) і H (i) представляють відповідно модулі, що тестуються, тестуючі модулі, та суміжні для модуля, зіставленого вершині i;

X (i) = {j ∈ V: (j, i) ∈ E}, Y (i) = {j ∈ V: (i, j) ∈ E}, H (i) = X (i) ∪ Y (i) і X (i) ∩ Y (i) = Ø.

Згідно ваги дуг, інцидентних вершині i, маємо:

X (i) = X0 (i) ∪ X1 (i), Y (i) = Y0 (i) ∪ Y1 (i). При цьому X0 (i) ∩ X1 (i) = Ø, Y0 (i) ∩ Y1 (i) = Ø.

Надалі для спрощення викладу тексту поширюється термінологія ОС на граф, яким вона описана, і надалі використовуються терміни «стан графа» замість «стан системи», «справна (несправна) вершина» замість «справний (несправний) модуль», «результат тесту (i, j)» замість «вага дуги (i, j)» і т.п.

Нехай задані D1, t (N) -граф, Fk і σ (Fk). Кожній вершині i графа зіставимо мітку m (i) ∈ {α, 0, 1}. Якщо за результатами аналізу σ (Fk) стан вершини i не знайдений, то m (i) = α, якщо вершина i визнана справної або несправною, то m (i) = 0 або m (i) = 1 відповідно (такі стани називаємо фінальними , а самі вершини - самовизначеними). Сукупність відміток, так само як і процес їх визначення, називається розміткою графа [6].

Розмітка графа виконується по-кроково. Якщо на черговому кроці визначений фінальний стан вершини i, то в подальшому можливість самовизначення для вершин j ∈ H (i) розглядається на залишковій множині R (j), утвореній виключенням з H (j) вершини i (так само, як і інших суміжних вершин у фінальному стані).

Динаміка зміни стану графа в процесі діагностування характеризується залишковим чином несправностей Fk - f, де f = ∪i∈V f (i) і f (i) - множина суміжних з i несправних вершин, ідентифікованих на попередніх кроках. Кожній вершині i графа співвіднесені значення | Fk - f (i) | границі самовизначення і τ (i) величини коригування цієї границі [6].

Початкове значення границі самовизначення для всіх вершин дорівнює величині кратності несправностей t, оскільки апріорна інформація про потужність образу несправностей відсутня. Границя змінюється в міру встановлення фінального стану для нових вершин.

Стан системи знаходиться в результаті перевірки правил самовизначення, що рекурсивно виконуються для кожної вершини графа:

$$[|(X1(i)) \cup (Y1(i))| > (t - \tau(i))]\ m(i) := 1; \quad (1)$$

$$[|(X0(i)| \geq (t - \tau(i))]\ m(i) := 0; \quad (2)$$

$$m(i) := 0\ [\forall\ j \in Y0(i)\{m(j) := 0\}]\ \&\ [\forall\ j \in (X1(i) \cup Y1(i))\{m(j) := 1\}]; \quad (3)$$

$$m(i) := 1\ \forall\ j \in X0(i)\{m(j) := 1\}. \quad (4)$$

Тут m (i): = c позначає операцію присвоєння вершині i мітки фінального стану: c ∈ {0, 1}. Вирази (1) - (2) вказують на ознаки фінального стану вершини i, виходячи зі значень синдрому, що мають відношення тільки до неї самої. Вирази (3) - (4) є умовами встановлення фінального стану суміжних з i вершин, що є наслідком ідентифікації фінального стану вершини i відповідно до (1) і (2). Встановлення фінального стану вершин відповідно до виразів (3) - (4)

випливає з правил визначення результатів тестування для ПМЧ-моделі, відповідно до яких:

[Y0 (i) ⊂ (V - Fk) & [X1 (i) ∪ Y1 (i)] ⊆ Fk], якщо вершина і справна, і X0 (i) ⊂ Fk, якщо вершина і несправна.

Нехай для вершини і діагностичного циркулянта {j, k} ⊂ H (i). Розглядається орієнтований повний граф K3 з вершин {i, j, k}. Трикутником тестування називається орієнтований підграф графа K3, що не містить взаємно зворотніх дуг. Для вершини і діагностичного циркулянта множина трикутників тестування на множині {i, j, k} задається вибором дуг K3. В роботі [2] виділені два базових трикутника тестування, що описані наборами дуг графа: A1 = {(i, j), (i, k), (j, k)} і A2 = {(i, j), (j, k), (k, i)}. Показується, що всі інші трикутники тестування еквівалентні базовим з точністю до позначення вершин. Надалі вказані трикутники тестування позначаються перерахуванням вершин (i, j, k), маючи на увазі зазначену вище послідовність його ребер.

В роботі [2] показано, що для трикутника тестування A1 синдроми {a (i, j), a (i, k), a (j, k)} = 001 або {a (i, j), a (i, k), a (j, k)} = 010 [12, 13] відповідають несправній вершині і при будь-якому стані вершин j і k. У трикутнику тестування виду A2 синдром {a (i, j), a (j, k), a (k, i)} = 001 відповідає несправній вершині i, синдром {a (i, j), a (j, k) , a (k, i)} = 010 - несправній вершині j, а синдром {a (i, j), a (j, k), a (k, i)} = 100 [12] - несправній вершині k. Вершина, для якої відповідно до синдрому, що співвіднесений з трикутником тестування, можна вказати фінальний стан, називається самовизначеною, а сам трикутник тестування - визначальним.

З множини трикутників тестування, які можна побудувати на множині Vi і Hi () () = ∪, для огляду необхідні тільки ті, які забезпечують самовизначення вершини і. Безліч цих трикутників відповідає випадку, коли діагностування забезпечується при мінімумі інформації, яка аналізується для кожної вершини.

Для вершини j Hi ∈ () ідентифікація фінального стану забезпечується з аналогічних трикутників тестування, побудованих на множині j ∪ H (j). Тому сказане вище резюмується у вигляді наступних формальних відношень:

{a(i, j), a(i, k), a(j, k)} = {001, 010} m(i):= 1;

(5) {a(i, j), a(i, k), a(j, k)} = {100} m(i):= 1. (6)

Трикутник тестування виду A1 будується на вершинах з i ∪ Y (i) і може бути використаним для самовизначення вершин при будь-яких значеннях N. Область використання трикутника виду A2 обмежена значеннями 21 3 t Nt + ≤ ≤, оскільки він утворюється на множині i ∪ X i ( ) ∪ Y (i) [6].

Визначення 1. Остаточний образ несправностей Fk - f називається самовизначеним, якщо і тільки якщо при будь-якому спільному з ним синдромі хоча б для однієї вершини з V - f виконуються умови самовизначення (1) - (6). В іншому випадку образ називається тупиковим.

Визначення 2. Граф називаємо локально t-діагностованим, якщо і тільки якщо для кожного Fk ∈ F (t) будь-який з його залишкових образів несправностей при Fk - f ≠ Ø є самовизначеним при будь-якому σ (Fk) [6].

За заданою кратності несправностей t область значень N, в якій можливе виникнення тупикових станів, обмежена значеннями N=2t+1 і деяким, емпірично визначеним, значенням Nb>N.Nb, є найбільше число вершин циркулянта, при якому ще існують такі образи несправностей потужності t, що для будь-якої справної вершини знайдеться хоча б одна несправна вершина, що її тестує. Описана властивість називається властивістю топологічного покриття графа.

Далі вказується, що несправна вершина j 1-покриває (0-покриває) вершину i, якщо a (j, i) = 1 (a (j, i) = 0). Образи несправностей, що топологічно покривають граф, можуть породжувати синдроми, при яких для кожної справної вершини і знайдеться 1-п несправна вершина j (умова тупикового

стану для справної вершини), а для кожної несправної вершини число інцидентних дуг з вагою 1 не перевищує поточного граничного значення (умова тупикового стану для несправної вершини). В роботі [1] значення Nb характеризується як найбільше число вершин циркулянта ступеня t, при якому виконується наступне:

$$\max \{ ( )\} (,) k b k F FN t b j F j Nt \in \in \sum \Delta \geq - . (7)$$

Тут Δ (j) - число несправних вершин, що тестують j, а Nb - t - число справних вершин графа.

При виконанні нерівності (7) знайдуться такі образи несправностей і спільний з ними синдром, що кожна справна вершина є 1-покритою.

При аналізі умов тупикового стану несправної вершини використовуються лема 1 і наслідок 1, які дають деякі загальні властивості образу несправностей, що топологічно покриває граф.

Лема 1. Для кожної вершини, при $N \geq 2t + 1$, $j \in Fk$ число справних вершин, що тестуєтсья, в Y (j) більше, ніж значення Δ (j) [6].

Доведення. Відповідно до конструкції циркулянта, | Y (k) | = | X (k) | = T для кожного $k \in V$. Число несправних вершин в Y (j) становить $\delta (j) \leq t - (\Delta (j) + 1)$ за умовою про кратність несправностей (рівність має місце при $N = 2t + 1$ і непарному N). Отже, число справних вершин в Y (j) буде $t - \delta (j) \geq \geq t - (t - (\Delta (j) +1)) = \Delta (j) +1$, що і завершує доведення.

Наслідок 1. Для кожної вершини $j \in Fk$ знайдеться хоча б одна 0-покрита справна вершина.

Грунтуючись на відношеннях (5) - (6), в найзагальнішому вигляді ознаки тупикового стану несправної вершини можна сформулювати в наступному вигляді.

Властивість 1. Вершина $j \in Fk$ має тупиковий стан, коли для кожного трикутника тестування (j, k, m) виду A1 виконується хоча б одна з умов

а) a (j, k) = 1,

б) a (k, m) = a (j, m) при наступній додатковій умові | Y1 (j) | ≤ Δ (j).

Властивість 1 виражається у вигляді сукупності ознак, що враховують взаємне розташування справних і несправних вершин множини Y (j).

Властивість 2. Аби для несправної вершини j досягався тупиковий стан, необхідно:

а) всі 1-покриті з j справні вершини i ∈ (V - Fk) ∩ Y (j) мають передувати 0-покритим справним вершин;

б) всі несправні вершини j1 ∈ Y (j), для яких a (j, j1) = 0, мають передувати 0-покритим справним вершин;

в) все несправні вершини j1 ∈ Y (j), які слідують за 0-покритими справними вершинами, мають a (j, j1) = 1;

г) для кожної 0-покритої несправної вершини j1 ∈ Y (j) значення a (j1, k) збігається зі значенням a (j, k), де k ∈ Y (j) ∩ Y (j1), при довільному (справний / несправне) стані вершини k [6].

Нехай для заданого образу несправностей Fk і спільного з ним синдрому для кожної вершини графа умови (1) - (4) самовизначення не виконуються. Це означає, що для кожної справної вершини і множина X1 (i) ≠ Ø, а для кожної несправної вершини j виконується умова | X1 (j) ∪ Y1 (j) | <T. Розглядаються умови тупикового стану несправної вершини, що випливають з невиконання для неї співвідношень (5) - (6) самовизначення за допомогою трикутників тестування виду A1.

Лемма 2. Щоб несправна вершина j не мала трикутників тестування, що визначають її стан, необхідно виконання наступних умов: ∃ j1 ∈ Y (j) ∩ Fk {a (j, j1) = 1}.

Доведення. Припускається протилежне: ∀j1 ∈ Y (j) ∩ Fk {a (j, j1) = 0}, але j не має трикутників, що визначають її стан. З властивості 2, випливає, що всі

1-покриті з j справні вершини i1 повинні передувати несправній вершині j1, найбільш віддаленої від вершини j зі зв'язків s1. З леми 1 випливає, що існує 0-покрита з j справна вершина i1> j1, і i1 = (j + t) mod N. З властивості 2-г випливає, що ∀j1 ∈ Y (j) ∩ Fk {a (j1 , (j + t)) = 0}, так як a (j, (j + t)) = 0 за умовою. Але вершина (j + t) може бути 1-покрита тільки з вершин безлічі {j ∪ (Y (j) ∩ Fk)}. Отже, при введених умовах порушується умова 1-покриття графа. Значить, при відсутності 1-покритих з j несправних вершин тупиковий стан вершини j не досягається. Кінець доведення.

Наслідок 2. Якщо несправна вершина j має тупиковий стан, то X0 (j) ≠ Ø.

Доведення. Нехай несправна вершина j 1-покрита усіма j1 ∈ X (j) ∩ Fk, тоді X0 (j) = Ø, тобто | X1 (j) | = T, і при | Y1 (j) | ≥ t (що має місце при 1-покритті з j несправних вершин) вершина j самовизначена за умовою (1).

Наслідок 3. Якщо для несправної вершини j при тупиковому стані графа виконується Δ (j) = 1, то вершина j має трикутник тестування, що її визначає.

Для аналізу локальної t-діагностованості графа, при заданому образі несправностей Fk, використовуються властивості трикутників тестування, побудованих на вершинах {j, k, (j + t)}, де (j + t)> k> j і j ∈ Fk.

Відповідно до наслідку 1, існує k ∈ V - Fk, для якого a (j, k) = 0. Якщо а) (j + t) ∈ V - Fk і a (j, (j + t)) = 1 або б) (j + t) ∈ V - Fk і a (j, (j + t)) = 1, то відповідно до (5) трикутник тестування (j, k, (j + t)) визначає j. Звідси необхідна умова, що для j ∈ Fk немає трикутників тестування, що визначають їх стан, має наступний вигляд:

якщо (j + t) ∈ V – Fk, то a(j, (j + t)) = 0; (8)

якщо (j + t) ∈ Fk, то a(j, (j + t)) = 1. (9)

У випадку (8) умова 1-покриття графа вимагає, щоб існувала така несправна вершина в Y (j), яка 1-покриває справну вершину (j + t). Без порушення умови, що j має тупиковий стан, такий вершиною може бути тільки

несправна вершина j1, 1-покрита з j. Це означає, що досягнення тупикового стану для j зменшує принаймні на дві одиниці загальну суму M (Fk) = () k j F j g ∈ Σ Δ -. Величина M (Fk) дорівнює найбільшому числу одиничних випадків тестування в синдромі, спільному з заданим образом несправностей, яка може бути використана для покриття несправних вершин графа без порушення умови про відсутність самовизначених несправних вершин. Тут g - число справних вершин в графі: g = N - | Fk |.

Для випадку (9) умова тупикового стану вершини j означає, що кожна несправна вершина j1 з Y (j) повинна бути 1-покрита з вершини j або сама повинна 1-покривати вершину (j + t). Таким чином, в разі (9) досягнення тупикового стану для j зменшує принаймні на 2δ (j) одиниць значення M (Fk).

Запропоновано алгоритм, який використовує описані умови для визначення таких сукупностей fk ⊆ Fk несправних вершин, тупиковий стан яких не порушує умови () () kk jF jfjg qj ∈ ∈ Σ Σ Δ -≥, що істотно зменшує складність аналізу локальної діагностованості, при заданому образі несправностей Fk. Тут q (j) - число одиничних випадків тестування, які використовуються для 1-покриття суміжних вершин при тупиковому стані вершини j [6].

З визначення 1-покриття графа випливає наступне

Наслідок 4. Якщо () k j F g j ∈ = Δ Σ, то всі несправні вершини j ∈ Fk мають трикутники тестування, що їх визначають виду A1.

Більш того, можна довести наступне.

Лема 3. Якщо для заданого образу несправностей Fk ∈ F (t) виконується () k j F j ∈ Σ Δ ≤ g + 4, то образ несправностей локально t-діагностований.

Лемма 3 дозволяє встановити нову нижню границю числа вершин локально t-діагнованого графа. Fkm позначається образом несправностей, для якого досягається максимальне значення суми M (Fk) на множині F (N, t).

Твердження 1. Якщо для заданих N і t при максимальному образі несправностей Fkm виконується () j Fkm j ∈ Σ Δ ≤ g + 4, то граф локально t-діагностований.

Твердження 1 має скоріше теоретичне, ніж практичне значення, оскільки для оптимальних графів вже при t> 6 має місце () j Fkm j ∈ Σ Δ> g + 4.

Разом з тим твердження 1 дозволяє зменшити число образів несправностей, що вимагають дослідження на можливість тупикового стану графа при () j Fkm j ∈ Σ Δ> g + 4. Тим самим зменшується складність вирішення задачі аналізу локальної t-діагностованості заданого графа.

Принципову можливість ефективного використання трикутників тестування демонструє наступне.

Лемма 4. В оптимальному циркулянтному графі для кожної несправної вершини є трикутник тестування виду A2 [6].

Доведення. За структурою, в циркулянті при N = 2t + 1 для будь-яких k, k1∈ V, таких що k1 ∈ Y (k), має місце Y (k1) ∩ X (k) ≠ Ø. Якщо k1 = k + a, то | Y (k1) ∩ X (k) | = A. Розглядається j ∈ Fk та приймається, що для неї немає жодного визначального трикутника тестування виду A1. Позначимо і справну вершину в Y (j), i = j + k, яка є 0-покритою справної вершиною, найбільш віддаленої від j зі зв'язків з міткою s1.

Якщо вершина j має тупиковий стан, то для будь-яких i1, i2 ∈ [Y (j) ∩ (V - Fk)], таких, що i1> i2, має місце {a (j, i1) ≥ a (j, i2) }. Отже, якщо i - найбільш віддалена від j 0-покрита справна вершина, то жоден трикутник тестування за участю вершини і не визначальним для j, якщо і тільки якщо {(i + 1), (i + 2), ..., (j + t)} ⊂ Fk. З урахуванням аксіоми про кратності несправностей звідси випливає існування i1 ∈ Y (i) ∩ X (j) ∩ (V - Fk); трикутник тестування (j, i, i1) є визначальним для вершини j.

Якщо в оптимальному циркулянті N = 2t + 2, то для кожної вершини k ∈ V вершина з номером (k + t +1) не входить в Y (k) ∪ X (k). Розглядається вершина j ∈ Fk, для якої відсутні визначальні трикутники тестування виду A1, і вершину i ∈ Y (j) ∩ (V - Fk), що є найбільш віддаленою від j зі зв'язків з міткою s1 0-покритою справною вершиною. Приймається, що i = j + k. З аксіоми кратності несправностей випливає, що загальне число несправних вершин в Y (i) не перевищує величини (t - 1). За визначенням, всі вершини, яким передує i, несправні, так що в Y (i) - Y (i) ∩ Y (j) міститься не більше (t - k - 1) несправних вершин. Припускається, що має місце найгірший випадок, тобто | [Y (i) - Y (i) ∩ Y (j)] ∩ Fk | = T - k - 1, тобто [X (j) - X (j) ∩ Y (i)] ∩ Fk = Ø.

Мають місце два випадки.

1. Якщо справна вершина в розглянутій множині Y (i) - Y (i) ∩ Y (j) має номер i1> j + t + 1, то існує трикутник тестування (j, i, i1), що визначає вершину j.

2. Нехай тепер i1 = j + t + 1. Необхідна умова тупикового стану вершини j - наявність Y1 (i) ≠ Ø; отже, є хоча б одна вершина j1 ∈ Fk ∩ Y (i) ∩ X (j), така, що a (j1, j) = 0, в такому випадку трикутник тестування (j, i, j1,) визначає j.

Наслідок 5. Оптимальний циркулянтний граф є локально t-діагностованим при використанні трикутників тестування.

Отже, розглянуто системний рівень діагностування ВС при множинних відмовах та наслідки тестування, відповідних ПМЧ-моделі. Як діагностична структура, був використаний циркулянтний граф з вершинним ступенем, що дорівнює кратності допустимих відмов t. Виявлено та проаналізовано умови, при яких локальне діагностування є неповним. Виведено умови достовірної ідентифікації стану модуля ОС, виробленої лише за результатами тестування з модулів, що мають з ним безпосередній зв'язок - умови локального t-діагностування, за допомогою порівняльного аналізу фрагментів синдрому,

співвіднесених зі структурною одиницею діагностичного циркулянта, названого трикутником тестування.

Аналіз показав, що для переходу вершини в тупиковий стан внаслідок відсутності визначальних її трикутників тестування необхідне одночасне виконання численних умов, пов'язаних як з взаємним розташуванням і числом суміжних несправних вершин, так і зі значенням фрагмента синдрому, який співвіднесений з даною вершиною. Одне це свідчить про те, що частка образів несправностей, які можуть призводити до тупикового стану графа, при використанні трикутників тестування значно скорочується. При цьому умови досягнення тупикового стану одними несправними вершинами можуть служити умовами самовизначення для інших вершин, в тому числі справних, що в кінцевому рахунку призводить до визначення стану всіх вершин графа. Численні приклади показали, що використання трикутників тестування у всіх розглянутих випадках забезпечує локальну t-діагностованість графа без використання додаткового тестування. Це і доведена ефективність трикутників тестування для локальної діагностованості оптимального циркулянта дозволяють висунути гіпотезу, що відношення (1) - (6) складають достатні умова для досягнення локальної t-діагностованих графа при будь-яких значеннях N і t.

Разом з тим аналіз показав, що спільний розгляд трикутників тестування, побудованих на множині $j \cup H(j)$, не дає додаткової інформації про визначення вершин, крім тієї, що може бути отримана зі співвідношень (4). Це свідчить про те, що для отримання нових умов визначення вершини необхідно розширювати окіл, що аналізується.

2.2.3. Метод організації самодіагностування багатопроцесорних систем з регулярною структурою

Обмеження 1. Граф діагностичних зв'язків має назву конструктивно регулярного. Цей граф має однотипні, а кожна вершина має 4 інцидентні дуги: 2 вхідні та 2 вихідні (рис.2.9) [3, 7].

Графи подібної структури називаються циркулянтами. Накладаються певні обмеження на граф: розглядаються лише однонаправлені орграфи, в яких не виконується тотожність $n=ki$, $n=kj$, $i=ki$, $j=ki$, де $k>1$ – ціле [7].

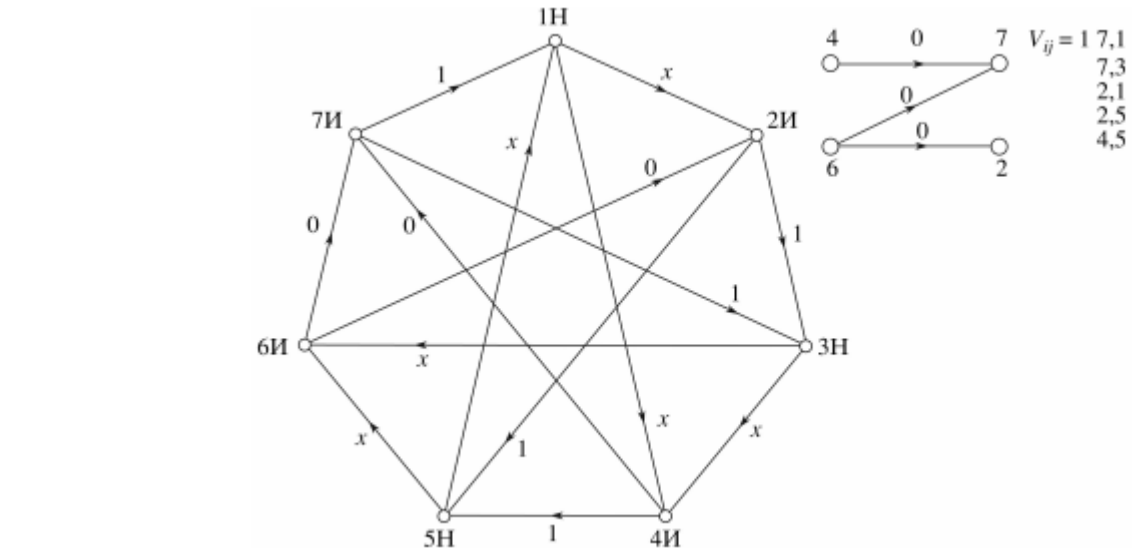


Рис.2.9 - Приклад графу G (1,3)

Обмеження 2. Максимальна кількість T несправних модулів (процесорів) обмежена. Розглядаються лише випадки $T \leq 4$.

Обмеження 3. Як модель несправностей, в методі використовується модель Препарата-Метца-Чена.

На дуги графа діагностичних зв’язків наносяться результати взаємотестування модулів системи у відповідності до ПМЧ-моделі. Для ПМЧ-моделі число несправних процесорів не перевищує значення $T<(n-1)/2$ [4].

Положення 1. Якщо на графі є дуга з результатом тестування $v_{ij} = 1$, стверджується, що хоча б один з процесорів i або j є несправним.

Положення 2. Якщо на діагностичному графі є 0-ланцюжок (рис.2.9) довжиною d +1, модулів, але система не допускає більше, ніж d відмов, то останній модуль справний[2].

Якщо цей модуль несправний, то несправними є всі інші модулі ланцюжка, що суперечить умові про максимальне допустиме число несправних процесорів у системі.

Визначення 1. Вагою P модуля називається число модулів, включаючи модуль, що розглядається, які підходять до нього 0-ланцюжком. Вага модуля поза 0-ланцюжком дорівнює одиниці [7].

Якщо в системі допускається m відмов, а після виконання всіх взаємоперевірок у діагностичному графі, утворився 0-ланцюжок, і останній модуль цього ланцюжка M має вагу P. Визначено, що серед інших модулів є L пар процесорів, що не перетинаються, і які відповідають положенню 1.

Положення 3. Якщо вага $P > m - L$, то модуль M справний.

В інакшому випадку загальне число несправних модулів перевищило б величини m.

Так як в процесі діагностики системи беруть участь, і справні, і несправні модулі, слід відзначити наступне.

Положення 4. Якщо стан процесора j встановлено, але результат перевірки v_{ij} суперечить йому, процесор i – несправний [7].

Суть методу полягає в наступному: після виконання всіх можливих перевірок виконується аналіз з використанням зазначених вище положень. Аналіз, виконується в декілька етапів. Кожний етап полягає у визначенні справного процесора з урахуванням зазначених вище обмежень та положень, а також визначенні станів модулів, які тестуються визначеним справним модулем. Аналіз виконується послідовно, і починається з визначення нульових ланцюжків. Слід відмітити, що справні модулі знаходяться з використанням

положень, а несправні визначаються за результатами тестування справними [3].

Розглядається випадок $T=3$ та $n\geq 7$.

Твердження 1. Існують такі конструктивно регулярні діагностичні графи, які не потребують, крім основних перевірок, більше однієї умовної перевірки для визначення стану процесорів системи, що має наступні параметри $T = 3$ та $n\geq 7$.

Існування таких графів можна довести на прикладі діагностичного графа $G(1, 3)$, який зображеного на рис.2.1. Буквою Н позначені несправні процесори, а I – справні, результати тестування, вказані на дугах [7].

Проводиться аналіз усіх перевірок на прикладі графа на рис.2.9. $P_7 = 3$, що означає: 7 = I, оскільки наявний одиничний результат перевірки v_{25}. Отже, 1=Н та 3=Н. 2 = I(за третім положенням), оскільки $P_2 = 2$, а лише один процесор із тих, що залишилися, може бути несправним. Результат тестування $v_{25} = 1$, він достовірний так, як другий процесор справний, отже, 5 = Н. Отже, всі несправні (1,3,5) модулі були встановлені, тоді інші модулі (2,4,6,7) справні. Діагностування повне, невизначеності не з'явились.

Аналіз не враховував результати типу x. Реальні значення x-в спотворити результати не можуть. Проте вони можуть покращити висновки аналізу у відповідності до положення 4.

Необхідно зазначити, що граф (рис.2.9) має дві вершини xx – обидві вхідні дуги мають мітку x (4 та 6 вершини). Отже, при тестування процесора двома несправними, його стан можна визначити.

Взаємне розташування справних та несправних процесорів представляється двійковим вектором, в якому 0 та 1 позначають відповідно несправний та справний стан. Для рис.2.9, двійковий вектор матиме наступний вигляд:

1 1 0 1 0 1 0

7 6 5 4 3 2 1

Для конструктивно регулярних графів, що розглядаються в цьому методі, кожному такому двійковому вектору відповідає *n* векторів, що отримуються циклічним зсувом. Аналізи діагностичного експерименту за графами, які відповідають цим векторам аналогічні, тому окремо вони не розглядаються.

Кількість векторів *R*, які можуть привести до різних результатів діагностування при *T* несправних процесорах та взаємно простих значеннях *T* та *n*, визначається наступною формулою:

$$R=\frac{C_7^3}{7}=5$$

, для *n* = 7 їх - 5(Рис.2.10).

1	1	1	1	1	0	0	0
2	1	1	1	0	1	0	0
3	1	1	0	1	1	0	0
4	1	0	1	1	1	0	0
5	1	1	0	1	0	1	0
	7	6	5	4	3	2	1

Рисунок 2.10 – Вектори розташування справних та несправних процесорів для *T* =3 та *n* = 7

Розглядається вектор 4:

1 1 1 0 0 1 0

7 6 5 4 3 2 1

Він призведе до появи вершини типу *xx*(вершина 4) і з високою ймовірністю до появи невизначеності, оскільки обидва тестуючих (1 та 3) модуля та модуль, що тестується, є несправними (рис.2.11) [7].

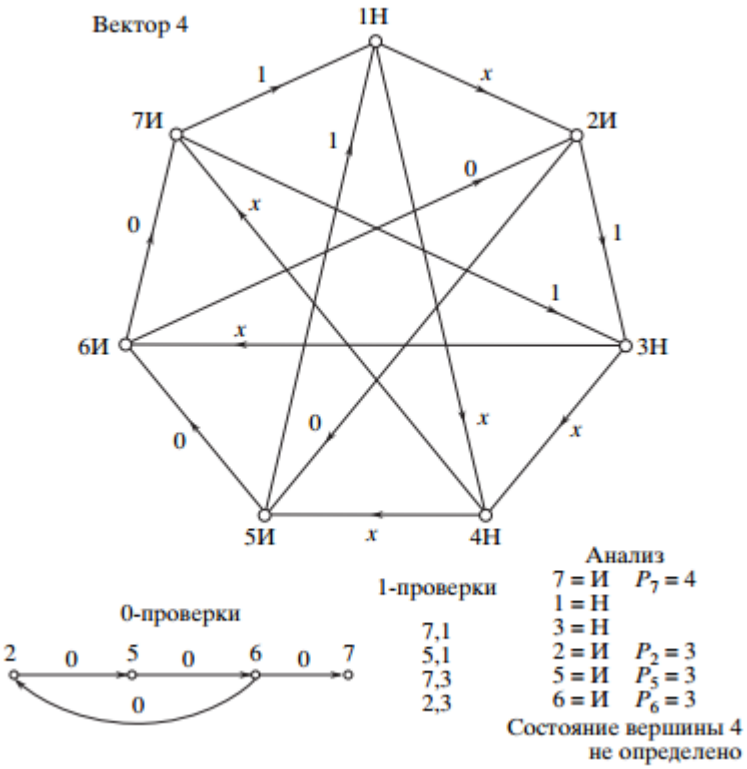


Рисунок 2.11 - Граф $G(1,3)$, вектор 4

Проводиться аналіз: процесор 7 справний за другим положенням, оскільки вага $P_7 = 4$. Визначаються також стани модулів 1 та 3: 1 = Н та 3 = Н за результатами тестування модулем 7. Так як серед процесорів, стан яких залишився невизначеним, може бути лише один несправний за умовою ПМЧ-моделі, то модулі 2, 5 та 6 справні за третім положенням [4]. Стани всіх процесорів, крім одного, визначені, а для визначення стану процесора 4 необхідна ще одна умовна перевірка.

Якщо реальне значення тестування дорівнює $v_{45} = 1$, $v_{47} = 1$, а, отже, результати тестів помилкові, то невизначеність зникає за 4 положенням: четвертий процесор несправний. Додаткова перевірка виявляється непотрібною.

Для інших векторів при діагностичному аналізі невизначеність не виникає. Має місце повне діагностування.

2.1.3. Випадок T=4

Розглядаються БС, які допускають кількість несправних процесорів T = 4 при n = 9. Варіанти векторів зображені на рис.2.12. Всього є 14 неповторюваних двійкових векторів взаємного розташування модулів. Розглядаються лише вектори які, відображають всі можливі результати тестування, в тому числі і виникнення невизначеностей[7].

	Розряди										Розряди										Розряди								
V	9	8	7	6	5	4	3	2	1	V	9	8	7	6	5	4	3	2	1	V	9	8	7	6	5	4	3	2	1
1	1	1	1	1	1	0	0	0	0	6	1	1	1	1	0	0	1	0	0	11	1	1	0	1	0	1	1	0	0
2	1	1	1	1	0	1	0	0	0	7	1	1	1	0	1	0	1	0	0	12	1	0	1	1	0	1	1	0	0
3	1	1	1	0	1	1	0	0	0	8	1	1	0	1	1	0	1	0	0	13	1	0	1	0	1	1	1	0	0
4	1	1	0	1	1	1	0	0	0	9	1	0	1	1	1	0	1	0	0	14	1	1	0	1	0	1	0	1	0
5	1	0	1	1	1	1	0	0	0	10	1	1	1	0	0	1	1	0	0										

Рисунок 2.12 - Вектори розташування процесорів С та Н для T=4, n =9

Твердження 2. Існують конструктивно регулярні діагностичні графи такі, що для встановлення стану модулів системи з параметрами n>=9, T = 4 достатньо, окрім основних перевірок, не більше двох додаткових.

Розглядається граф G(3,4) та вектор 1(рис.2.12). Граф зображений на рис.2.13. Таке розташування справних та несправних процесорів призводить до появи 3 вершин типу *xx* (5, 6, 7), але система залишається повністю діагностованою. На 1-у етапі визначаються 9 = I за третім положенням так, як вага P_9=3, також мають місце 1-перевірки $7 \rightarrow 2$ та $8 \rightarrow 3$. За властивостями моделі Препарата-Метца-Чена процесори 3 та 4 несправні [4]. На 2-у етапі можуть бути лише 2 несправних процесора, визначається стан 8-о процесора – він справний, оскільки вага P_8=2 та результат тестування v_{61}=1. Отже, другий процесор несправний, а вершини, стан яких ще не визначений, можуть включати в себе тільки один несправний процесор. На третьому етапі визначається стан 6-о модуля – справний, оскільки v_{71}=1, перший процесор несправний. Всі стани процесорів встановлені, діагностування повне.

У випадку для чотирьох відмов реальні *x*-дуг не можуть спотворити результат.

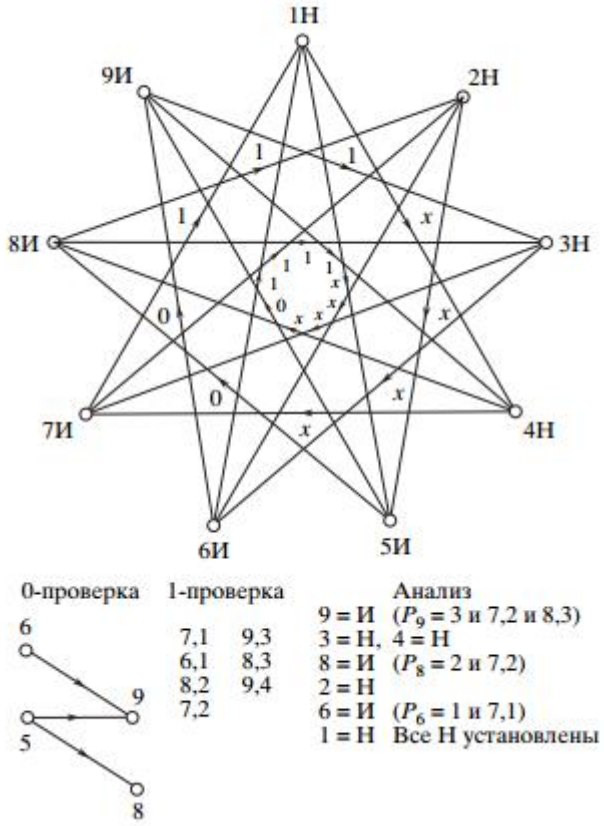


Рисунок 2.13 - Граф *G*(3,4), аналіз для вектору 1

Зазначається, що одним із найгірших векторів розміщення процесорів є вектор 2 (рис.2.12). При діагностуванні графа для цього вектору невизначеними залишаються вершини 5 та 6. В деяких випадках необхідно 2 додаткові перевірки для визначення їх стану. Як і у випадку з T=3, реальні значення x-дуг можуть тільки покращити результат діагностування.

2.3. Висновки

У даному розділі розглянуто та обґрунтовано вибір моделі несправності та проаналізовані існуючі методи організації самодіагностування багатопроцесорних систем.

Як модель несправностей, було обрано модель Препарата-Метца-Чена, оскільки дана модель має найбільш широке використання у дослідженнях, пов’язаних з багатопроцесорними системами та не накладає обмежень на результат тестування несправним процесором іншого модуля.

Вона заснована на наступних передумовах: система розбивається на модулі, кожен з яких може самостійно тестувати інший модуль; тестуючий модуль оцінює стан модуля, що тестується; результат тесту завжди достовірний, якщо тестуючий модуль – справний, і недостовірний - в іншому випадку. Результати перевірки системи утворює двійковий вектор $R = r1 \ldots rl$, який називається реальним синдромом [4].

Розглянуті методи самодіагностування використовують модель Препарата-Метца-Чена, як модель несправностей.

Метод, описаний у пункті 2.2.1, побудований на використанні системи тестів, що відповідають p ланцюжках певної структури, побудованим для перевірки кожного з $n \geq (2p+1)$ модулів системи. Розроблений метод дешифрування реального синдрому, отриманого при перевірці системи на даній системі тестів, що включає корекцію вихідного синдрому для відсіювання недостовірних реакцій, отриманих при використанні в ланцюжках контролюючих несправних модулів, і гарантує достовірність результатів вторинного діагнозу. У ряді випадків при дешифрування скоригованого синдрому можуть виявитися невизначеними технічні стани деяких справних модулів [5].

У методі, що був описаний у пункті 2.2.2, використовуються трикутники тестування. Розгляд трикутників тестування не дає додаткової інформації про визначення вершин, крім тієї, що може бути отримана зі співвідношень, описаних у методі. Це свідчить про те, що для отримання нових умов визначення вершини необхідно розширювати окіл, що аналізується [6].

У цих методах виникає алгоритмічна та часова надлишковість, завдяки кількості перевірок, а також часу, який необхідний на проведення аналізу результатів тестування. Тому за основу для розробки методу скорочення часу самодіагностування багатопроцесорних систем було обрано метод, описаний у пункті 2.2.3. Метод організації самодіагностування багатопроцесорних систем з регулярною структурою при невеликій кількості тестових зв’язків та числом взаємоперевірок дозволяє повністю визначити технічний стан системи, за винятком одного або двох процесорів в деяких випадках [3, 7].

3. МЕТОД СКОРОЧЕННЯ ЧАСУ САМОДІАГНОСТУВАННЯ У БАГАТОПРОЦЕСОРНИХ СИСТЕМАХ

3.1. Опис методу зменшення часу самодіагностування у багатопроцесорних системах

Як було вказано у висновках до попереднього розділу за основу для розробки методу обрано модель Препарата-Метца-Чена, як модель несправностей та метод організації самодіагностування з регулярною структурою. Нагадаємо основні положення моделі Препарата-Метца-Чена:

- система розбивається на модулі так, що кожний з модулів може тестувати інший модуль ;
- система представляється у вигляді діагностичного графа, в якому модулі системи представлені вершинами графа, а діагностичні зв'язки між процесорами – дугами;
- модель накладає наступні умови на результат тестування (v_{ij}) i-м процесором j-го: якщо i та j обидва справні, то v_{ij}=0; якщо процесор i - справний, а процесор j – несправний, то v_{ij}=1; якщо i-й процесор несправний, то результат тестування $v_{ij}=x$, де x приймає значення 0 або 1 незалежно від стану процесору, що тестується [4, 7];
- модель накладає обмеження на допустиму кількість несправностей у системі - $n/2 > T$, де n позначає кількість процесорів, а T - кількість процесорів, що відмовили [4].

Метод використовує діагностичні графи, які описані у пункті 2.2.3, графи названі конструктивно регулярними і мають наступні характеристики та обмеження:

- кожна вершина має чотири інцидентні дуги, 2 вхідні та 2 вихідні;
- використовуються лише однонаправлені графи;

- накладаються обмеження на n – кількість процесорів, t – допустиму кількість відмов та стрибки графа i та j: $n \neq ki,\ n \neq kj, i \neq kj,\ j \neq ki, k > 1$, при цьому k – ціле [3, 7].

Головною особливістю запропонованого метода є спосіб, в який проводиться діагностування системи. Пропонується послідовне діагностування системи. В літературі «послідовним діагностуванням» прийнято називати спосіб організації діагностування, при якому після знаходження модуля, що відмовив, проводиться його заміна на справний, а процес діагностування завершується після визначення всіх модулів, що відмовили [8]. У запропонованому методу під «послідовним діагностуванням» розуміється наступне: в кожний момент часу може тестуватися лише одна пара процесорів (інші процесори участі в тестуванні в цей момент не приймають), після отримання результату тестування проводиться аналіз та визначається стан модуля/модулей, якщо це можливо, та обирається наступна пара модулів для тестування. Діагностування системи завершується після визначення стану всіх її компонентів за винятком появи однієї або двох невизначеностей [8].

Аналіз і визначення стану системи проводиться на основі результатів тестування. Стани процесорів визначаються за допомогою побудови 0-ланцюжків. 0-ланцюжок – це послідовність вершин діагностичного графа, які з’єднані між собою дугами, результат тестування яких дорівнює «0» [8, 9].

Визначаються ваги P кожного модуля, який входить до ланцюжка. Вага модуля дорівнює кількості вершин, які підходять до цього модуля 0-дугами. Вага кожного модуля, що не входить до ланцюжка, дорівнює 1 [3, 7].

Якщо довжина 0-ланцюжка більша допустимого значення T, тоді вага останнього процесора дорівнює або більше значення T+1, і за положенням метода [7] та моделлю Препарата-Метца-Чена [4], цей модуль справний,

інакше несправними є всі процесори в цьому ланцюжку, тоді число несправних процесорів перевищує значення T, що суперечить умові.

Несправні модулі визначаються шляхом тестування справними. Кожний визначений несправний модуль зменшує допустиму кількість несправних процесорів в системі T на 1 [8].

Наведемо детальний алгоритм діагностування, який закладений в основу метода.

3.1.1. Алгоритм самодіагностування багатопроцесорної системи

При самодіагностуванні багатопроцесорної системи за допомогою запропонованого метода використовується наступний алгоритм.

1. Обирається пара модулів та проводиться тестування процесорів.

1.1. Якщо значення дуги дорівнює «0», то ця пара процесорів додається до 0-ланцюжка, визначаються ваги цих процесорів, перевіряється умова P>T.

1.1.1. Якщо умова виконується, останній процесор в 0-ланцюжку справний. Перехід до пункту 2.

1.1.2. Якщо вага модуля менше значення T, то обирається наступна пара процесорів, в якій тестуючим має бути процесор j з попередньої пари. Це необхідно для побудови ланцюжка.

1.2. Якщо результат тестування – «1», то дана пара процесорів виключається з тестування, оскільки хоча б один процесор з цієї пари є несправним, і результаті їх тестування ненадійні.

1.2.1. Перевіряється наступна умова: $R \geq T$, де R – кількість виключених з тестування пар процесорів, а T – допустима кількість відмов на момент тестування. Якщо умова виконується, то всі процесори, які не були виключеними з тестування, визначаються справними. Перехід до пункту 2.

2. Якщо знайдено хоча б один справний процесор, перевіряється всі його вихідні дуги, оскільки його стан визначено і результати тестування цього процесора вважаються надійними.

2.1. Якщо результат тестування «0», то процесор, що тестується, є справним.

2.2. Якщо =1, то процесор є несправним.

3. Перевіряються вхідні дуги процесора, стан якого було визначено. Якщо є відомий результат тестування вхідної дуги, та він не співпадає зі станом процесора, то і-й процесор несправний.

4. Перевіряється кількість визначених процесорів. Якщо кількість визначених процесорів дорівнює значенню n або кількість несправних визначених процесорів дорівнює значенню T, процес діагностування вважається завершеним.

4.1. Якщо було проведено 2n перевірок, а стан одного процесора при T=3 та двох процесорів при T=4 залишився невідомим, то це свідчить про появу невизначеності у системі і необхідна одна або дві додаткові перевірки для визначення стану цих процесорів.

Розглянемо приклад. Нехай параметри системи наступні:

- n=9, t=3, i=1, j=4,
- несправні процесори: 4, 5, 8.

На рис.3.1. зображено граф з заданими параметрами. Також на рисунку позначені результати тестування. Задаємо реальні значення тестування процесорів 4, 5, 8. Нехай $v_{45} = 0$, $v_{48} = 1$, $v_{56} = 1$, $v_{59} = 0$, $v_{89} = 0$, $v_{83} = 0$.

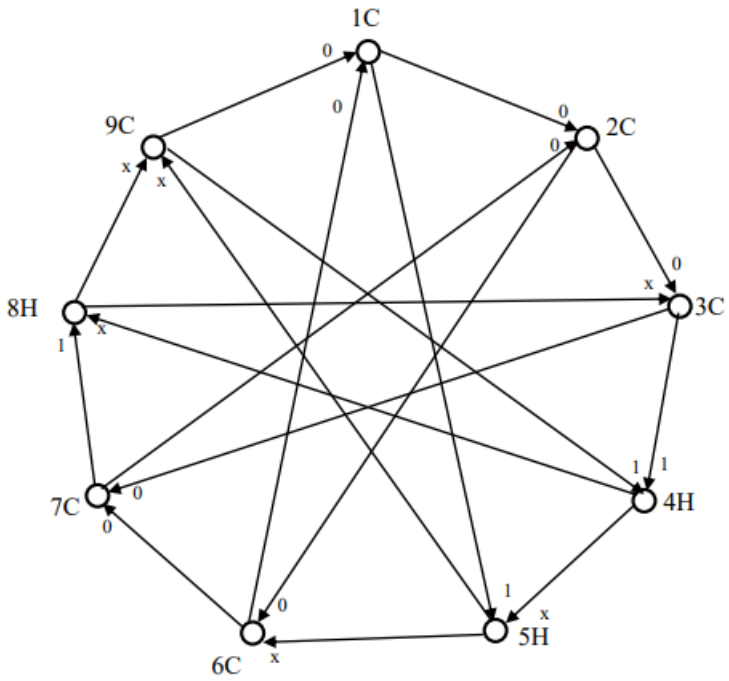
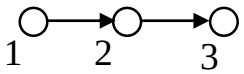


Рисунок 3.1 – Діагностичний граф

Тоді процес діагностування наступний: обираємо першу пару процесорів для тестування, нехай, це процесори 1 та 2 (m=1). Результат $v_{12} = 0$, додаємо пару у 0-ланцюжок, ваги P процесорів відповідно P1=1, P2=2.

Обирається наступна пара: процесори 2 та 3 (m=2). Результат тестування «0», пара процесорів додається до 0-ланцюжка:



Ваги модулів дорівнюють: P1=1, P2=2, P3=3. P3 менше, ніж T, тому обирається наступна пара, процесори 3 та 4 (m=3). Результат тестування $v_{34} = 1$, отже, хоча б один процесор є несправним, пара виключається з процесу діагностування. Отже, максимальна кількість несправностей дорівнює на одиницю менше, t=T-1=2. Обирається наступна пара, модулі 5 і 6 (m=4). Результат тестування «1». Вони також виключаються с процесу діагностування, t=1. Результат тестування наступної пари модулів(7 та 8) також «1» (m=5). Вони виключаються з діагностування, отже, t=0, з

діагностування були виключенні всі пари процесорів, в яких один модуль з пари точно є несправним, тому всі процесори, що залишилися, є справними. Отже, модулі 1, 2, 9 – справні.

Перевіряємо діагностичні зв’язки справних процесорів так, щоб виявити несправні процесори за найменшу кількість перевірок:

1. 2 → 3, m=6, $v_{23} = 0$, процесор 3 – справний. Отже, 4 – несправний.

2. 1 → 5, m=7, $v_{15} = 1$, процесор 5 – несправний.

3. 3 → 7, m=8, $v_{37} = 0$, процесор 7 – справний. Отже, 8 – несправний.

Знайдено всі несправні процесори (4, 5 та 8), отже, процесори, що залишилися – справні. Процес діагностування завершений за m=8 перевірок. Невизначеностей не виникло.

Запропонований метод дозволяє зменшити кількість взаємоперевірок при самодіагностуванні багатопроцесорних систем. Для оцінки верхньої та нижньої границі кількості взаємоперевірок при використанні методу, а також перевірки роботи методу, було створено програму, яка моделює процес самодіагностування багатопроцесорних систем з використанням запропонованого методу. Опис програмних засобів, які були використані при створенні програмної моделі, опис структури та роботи програми наведені у наступному пункті.

3.2. Структура та опис роботи програми самодіагностування

3.2.1. Опис використаних програмних засобів

Програмну модель було розроблено на мові програмування Python. Python - це високорівнева, інтерпретована та об'єктно-орієнтована мова програмування [10]. Загалом мова використовується для створення веб-додатків, автономних програм (за допомогою сторонніх інструментів), або програм, інтегрованих в більшу систему. Python підтримує функціональний, об'єктно-орієнтований та імперативний стилі кодування, що наділяє мову

певною універсальністю. Python має наступні особливості та переваги в порівнянні з іншими мовами програмування:

1. Якість програмного забезпечення. Код, написани на Python відрізняється читабельністю та зрозумілістю, оскільки мова має простий та лаконічний синтаксис, що полегшує написання коду та подальшу його підтримку. Крім того, Python підтримує механізми повторного використання програмного забезпечення, таких як об'єктно-орієнтоване та функціональне програмування [10].

2. Продуктивність розробника. Розмір програм на Python суттєво менший, ніж на таких мовах програмування, як C, C ++ і Java за рахунок знову ж таки простого синтаксиса. Програми на Python не потребують тривалої компіляції, що значно пришвидшує роботу.

3. Портативність програми. Більшість програм Python працюють без змін на всіх основних комп'ютерних платформах. Наприклад, перенесення коду Python між Linux і Windows зазвичай полягає у переносі коду між машинами. Крім того, Python пропонує кілька варіантів кодування портативних графічних користувацьких інтерфейсів, способів доступу до баз даних, веб-систем тощо [10].

4. Підтримка бібліотек. Python має велику кількість попередньо встановлених і портативних функцій, відомих як стандартна бібліотека. Ця бібліотека підтримує безліч завдань програмування на рівні програми, від структури до мережевих скриптів. Python пропонує інструменти для створення веб-сайтів, числового програмування, доступу до послідовного порту, розробки ігор та багато іншого. Наприклад, розширення NumPy було описано як безкоштовний і більш потужний еквівалент Matlab.

5. Інтеграція компонентів. Програми на Python можуть легко спілкуватися з іншими частинами програми, використовуючи різні механізми

інтеграції. Така інтеграція дозволяє використовувати Python, як інструмент налаштування та розширення продукту. Python може викликати бібліотеки C та C ++, може викликатися з C і C ++ програм, інтегруватися з Java та .NET компонентами, підтримувати зв'язок через такі системи, як COM та Silverlight, здатен взаємодіяти через мережі з інтерфейсами, такими як SOAP, XML-RPC і CORBA [10].

Для розроблення програми самодіагностування було обрано дану мову програмування, оскільки Python має широкий вибір математичних бібліотек, зокрема для роботи з графами, має простий синтаксис, що дозволяє швидко змінювати поведінку програми та є кросплатформенним.

Основною бібліотекою, яка використовувалася при розробці моделі стала бібліотека Networkx для роботи з графами. Бібліотека має наступні методи для роботи з графами:

- задання графів будь-якого типу;
- можливість додавати, видаляти, змінювати вершини графа с додаванням до них певних тегів, наприклад, для задання ваги вершини;
- набір функцій для керування набором дуг між вершинами;
- вбудовані функції для роботи з направленими графами;
- функціїї для роботи з мультиграфами;
- вбудовані методи для аналізу графу, наприклад, робота з підграфами, знаходження графів-циркулянтів тощо;
- методи для виведення графів, використовуючи бібліотеку matplotlib [11].

Для візуалізації графів та виведення даних, крім вбудованих бібліотек, була використана бібліотека matplotlib, що підтримує широкий спектр діаграм та графіків: звичайні графіки, стовпчасті та секторні діаграми, діаграми розсіювання, контурні графіки, поля градієнтів тощо. Matplotlib сумісна з

багатьма іншим бібліотеками, що суттєво спрощує необхідність візуалізації даних. Бібліотека часто використовується разом з numpy, pandas.

3.2.2. Структура програми самодіагностування

Програма самодіагностування складається с окремих модулів, кожний з яких має визначений набір функцій і є автономним по відношенню до інших модулів, що дозволяє досить легко змінювати поведінку програму, якщо це необхідно. Програма має наступні модулі:

1. Керуючий модуль. Відповідає за виклик інших модулів системи, отримує та перевіряє вхідні параметри для побудови діагностичного графа. Передає параметри в модуль побудови графа.
2. Модуль побудови діагностичного графа. Використовуючи параметри, отримані від керуючого модуля, а саме: кількість процесорів, допустима кількість відмов у системі, стрибки графа, несправні процесори(опціонально, програма може обрати несправні процесори випадково), реальні значення тестування несправних процесорів (опціонально, програма може згенерувати їх самостійно), будує діагностичний граф, наносить результати тестування.
3. Модуль самодіагностування системи. Реалізує роботу методу, викладеного в пункті 3.1. Здійснює тестування процесорів системи та проводить аналіз. Викликає модуль розрахунку ваги вершин. Після завершення діагностування та аналізу передає дані в модуль аналізу результатів.
4. Модуль побудови 0-ланцюжків. Виконує побудови 0-ланцюжків, розраховує ваги вершин, передає дані в модуль самодіагностування. Для спрощення сприйняття програми було вирішено винести цей функціонал з модуля самодіагностування в окремий модуль.

5. Модуль аналізу результатів. Приймає дані з модуля самодіагностування для перевірки правильності результатів діагностування, порівнює визначений стан процесорів з початковими параметрами, виводить інформацію про помилки діагностування, якщо вони є, та надає інформацію про стан системи.
6. Модуль виводу результатів. Приймає дані з модуля аналізу результатів, записує розгорнутий опис процесу самодіагностування у окремий файл, малює діагностичний граф та виводить основну інформацію про систему на екран користувача.

3.2.3. Опис роботи програми самодіагностування

На початку роботи ввести початкові дані такі, як: кількість процесорів в системі, допустима кількість відмов, стрибки графа, номери несправних процесорів, а також реальні значення тестування x-дуг. Введені параметри перевіряються, у випадку виникнення помилки, надається відповідне повідомлення. Слід зазначити, що програма надає можливість згенерувати ці дані автоматично без участі користувача.

Також необхідно обрати тип самодіагностування: одиничний або повний. У першому варіанті розглядається один конкретний випадок діагностування, у другому – при заданій кількості процесорів, відмов та стрибків, система послідовно перебирає всі можливі випадки розташування несправних процесорів та реальних значень x-дуг.

Вхідні дані передаються до модуля побудови діагностичного графа, в якому з використанням методів бібліотеки Networkx, будується граф (вершини, дуги), створюється вектор відмов та на дуги наносяться результати тестування.

Приклад запису початкових даних наведений нижче

```
Enter number of nodes:  7
Enter i-step: 1
Enter j-step: 3
Enter number of failures: 3
[1, 2, 3, 4, 5, 6, 7]
[(1, 2), (1, 4), (2, 3), (2, 5), (3, 4), (3, 6), (4, 5), (4, 7), (5, 6),
(5, 1), (6, 7), (6, 2), (7, 1), (7, 3)]
```

Приклад функції задання вершин графа:

```
def add_nodes(gr, n):
    gr.add_nodes_from(range(1, n+1))
    return gr.nodes(data=False)
```

Приклад запису результатів тестування з урахуванням положень ПМЧ-моделі:

```
def edges_state(gr, dict_p_s):
    dict_edges_real_state = {}
    for u, v in gr.edges():
        if dict_p_s['proc' + str(u)] == 0 and dict_p_s['proc' +
str(v)] == 0:
            dict_edges_state[(u, v)] = 0
        elif dict_p_s['proc' + str(u)] == 0 and dict_p_s['proc' +
str(v)] == 1:
            dict_edges_state[(u, v)] = 1
        else:
            dict_edges_state[(u, v)] = 'x'
    dict_edges_real_state = dict_edges_state.copy()
    for u, v in gr.edges():
        if dict_edges_real_state[(u, v)] == 'x':
           dict_edges_real_state[(u,v)] = random.randrange(2)
    return dict_edges_state, dict_edges_real_state
```

Після закінчення роботи модуль передає всі необхідні дані в модуль самодіагностування та починається процес визначення станів модулів системи згідно алгоритму, описаного в пункті 3.1.1.

Обирається пара процесорів для діагностування. Перевіряється, чи не обиралась ця пара для діагностування раніше. Якщо обиралась, то використовується попередній результат тесту.

```
if not((u, v) in dict_of_checked_edges):
    dict_of_checked_edges[(u, v)] = dict_real[(u, v)]
        dict_of_counters['Number of check'] += 1
```

Якщо результат тестування «1», то процесори виключаються з діагностування, та відповідна пара заноситься у словник виключених процесорів. Перевіряється лічильник виключених пар процесорів. Якщо кількість виключених пар дорівнює кількості несправних процесорів на момент тестування, то всі невиключені процесори автоматично вважаються справними.

```
processor_state['proc' + str(proc_1 + 1)] = 0
dict_of_counters['Number of established states'] += 1
```

Далі необхідно лише *t* перевірок справних процесорів для тестування одного процесора з кожної виключеної пари.

Якщо результат тестування «0», то до модулю побудови 0-ланцюжків передаються дані про id процесорів, які з’єднані 0-дугою. Модуль побудови 0-ланцюжків додає знайдену пара процесорів до дерева і визначає ваги кожного модуля.

```
zero_gr = nx.DiGraph()
    zero_gr.add_nodes_from(range(1, n + 1))
    zero_gr.add_edge(out_n, in_n)
```

Якщо до цього в модулі вже було існуюче дерево процесорів, з’єднаних дугами з результатами тестів «0», нові процесори додаються у дерево та всі ваги модулів в дереві обчислюються заново.

Ваги модулів обчислюються завдяки матриці зв’язків.

```
for i in range(0, n):
    for j in range(0, n):
        if nx.has_path(zero_gr, i + 1, j + 1) and i != j:
            matrix_of_connections[i][j] = 1
```

Обирається найдовший 0-ланцюжок та останній процесор в цьому ланцюжку. Дані передаються у модуль самодіагностування та проводить перевірка чи вага цього модуля більше, ніж допустиме число відмов в системі.

Якщо так процесор визначається як справний, та збільшується лічильник визначених процесорів та визначених справних процесорів:

```
if z['Maximum'] - 1 > t:
     processor_state['proc' + str(v)] = 0
     dict_of_counters['Number of established states'] += 1
     dict_of_counters['Number of established zero'] += 1
```

Якщо визначений стан хоча б одного справного процесора викликається рекурсивна функція перевірки його зв’язків. Функція припиняє роботу, якщо стан всіх процесорів визначено, знайдено всі несправні процесори або вже неможливо визначити стан жодного процесора.

У випадку, коли етап діагностування завершений, але залишились невизначені процесори, обирається наступна пара для тестування, жодний процесор з якої не бува виключений з процесу діагностування. Слід зауважити, що процес самодіагностування припиняється до явного визначення всіх процесорів тільки у випадку, коли знайдено максимальну допустиму кількість несправних процесорів *T*.

```
if d_o_con['Number of established unit'] == t:
  for i in range(n):
     if pr_st['proc' + str(i + 1)] is None:
        pr_st['proc' + str(i + 1)] = 0
        d_o_con['Number of established zero'] += 1
        d_o_con['Number of established states'] += 1
  break
```

У всіх інших випадках необхідно знайти всі справні процесори(оскільки при заданій кількості *T,* відмов у системі може бути і менше). Якщо було перевірено всі діагностичні зв'язки, а стан деяких процесорів так і залишився невідомим, модуль самодіагностування завершує свою роботу та передає дані у модуль аналізу результатів.

Цей модуль проводить аналіз даних, отриманих від модуля самодіагностування, порівнює їх з початковими даними, та виводить користувачу інформацію про стан системи, та надає інформацію про невизначеності, якщо вони виникли в ході процесу діагностування системи. Також модуль створює текстовий файл з докладним описом процесу діагностування. Нижче наведений приклад такого файлу с висновком про результат самодіагностування.

```
      dict of checked:  {(1, 2): 'x', (3, 4): 1, (5, 6): 1, (7, 1): 1, (7, 3):
0, (3, 6): 1}
      Initial processors states:
       {'proc1': 1, 'proc2': 0, 'proc3': 0, 'proc4': 1, 'proc5': 0, 'proc6': 1,
'proc7': 0}
      State of processors after diagnosis:
       {'proc1': 1, 'proc2': 0, 'proc3': 0, 'proc4': 1, 'proc5': 0, 'proc6': 1,
'proc7': 0}
      Number of check:  6
      Diagnosis succeed
```

3.3. Висновки

У даному розділі магістерської дисертації було запропоновано метод скорочення часу самодіагностування у багатопроцесорних системах, надано опис діагностичного графу, який використовується в методі, та алгоритму, надано приклад проведення процесу самодіагностування з використанням запропонованого методу.

Необхідно зазначити, що метод лише дає обмеження на діагностичний граф, а не структуру системи в цілому, тому в системах з більшою кількістю

зв’язків між модулями все ще можна використовувати запропонований метод, якщо виконуються необхідні умови для побудови графу. В такому випадку при діагностуванні будуть використовуватися не всі зв’язки системи, але при виникненні невизначеностей такі зв’язки можуть здійснювати додаткові перевірки для визначення процесорів, стан яких після процесу діагностування, залишився невідомим.

В ході роботи було створено програму самодіагностування багатопроцесорних систем з використанням запропонованого методу, обґрунтовано вибір програмних засобів, які використовувались при створенні програми, здійснено опис структури та роботи програми.

4. АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ ПРОГРАМНОЇ МОДЕЛІ

У минулому розділі було розглянуто запропонований метод організації самодіагностування багатопроцесорних систем. Необхідно провести аналіз результатів, отриманих в результаті роботи програми самодіагностування, визначити верхню та нижню границі кількості перевірок, порівняти запропонований метод з існуючими та описати випадки виникнення невизначеностей та описати особливості роботи аналізу програми діагностування.

4.1. Визначення верхньої та нижньої границі кількості перевірок

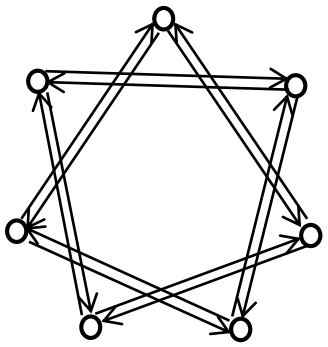
В ході роботи над магістерською дисертацією визначено верхню та нижню границі кількості перевірок при використання запропонованого методу.

4.1.1. Визначення нижньої межі кількості перевірок

Нижня границя кількості перевірок, при якій можна визначити стан системи, дорівнює значенню – T+1 [9]. Розглянемо обмеження, які накладаються на граф для досягнення нижньої межі кількості перевірок:

$$\begin{cases} i \in \{0; n-1\}, \\ j = n-1, \end{cases}$$

тобто обмеження накладаються лише на стрибки графа [8]. Розглянемо ситуацію більш детально: на рис.3.2 зображений граф та описана послідовність ідей, яка необхідна для визначення стану системи за T + 1 перевірки.



1. 1→3 $v_{13} = 0$. $\overset{1}{\circ}\!\!\rightarrow\overset{3}{}$
2. 3→5 $v_{35} = 0$. $\overset{1}{\circ}\!\!\rightarrow\overset{3}{\circ}\!\!\rightarrow\overset{5}{}$
3. 5→7 $v_{57} = 0$. $\overset{1}{\circ}\!\!\rightarrow\overset{3}{\circ}\!\!\rightarrow\overset{5}{\circ}\!\!\rightarrow\overset{7}{}$
4. 7→5 $v_{75} = 1$.

Рисунок 4.1. Нижня межа кількості перевірок

Послідовність дій визначення стан системи:

1. 1 → 3, v_{12} = 0, будується 0-ланцюжок. Вага процесорів 1 та 2 відповідно.
2. 3 → 5, v_{23} = 0, процесори 3 та 5 додаються до 0-ланцюжка. Вага процесорів P1=1, P3=2, P5=3.
3. 5→ 7, v_{57} = 0, процесори додаються до ланцюжка. Вага процесорів P1=1, P3=2, P5=3, P7=4, вага 7-о модуля P7 > T, отже, процесор 7 - справний.
4. 7→ 5, v_{75} = 1. Процесор 7 – несправний, а, отже, за ПМЧ-моделлю [4] несправні процесори 1 та 3. Всі несправні процесори визначено. Процесори 2,4,6 – справні. Діагностування завершено.

Стан системи визначено за T+1 = 4 перевірки.

4.1.2. Визначення верхньої межі кількості перевірок

Стверджується, що верхня границя кількості перевірок дорівнює значенню n+T за винятком випадків, коли необхідна одна або дві додаткові перевірки, у зв’язку з виникненням невизначеності [8]. Це означає, що в середньому для визначення справного модуля в системі необхідно виконати одну перевірка, а для визначення несправного – дві. При пошуку верхньої

границі кількості взаємоперевірок з використанням програми значення кількості перевірок не перевищувало значення n+T-2 [8].

Розглянемо приклад, який демонструє визначення стану системи за m=8 перевірок. Початкові параметри системи: n=8, T=3, i=1, j = 3. Несправними модулями є: 1, 3, 8 процесори. Реальні значення дуг наступні: $v_{12} = 1$, $v_{14} = 1$, $v_{34} = 0$, $v_{36} = 0$, $v_{81} = 1$, $v_{83} = 1$. На рис.4.2 зображений граф з заданими параметрами.

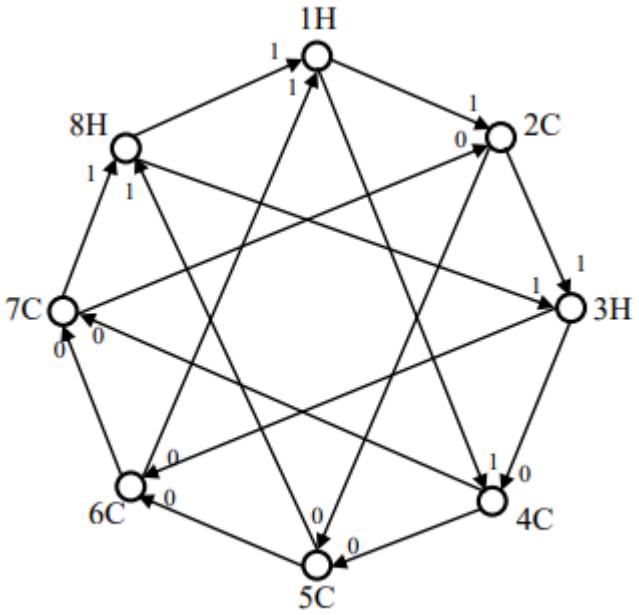


Рисунок 4.2 – Граф для системи з n=8 та t=3

Послідовність дій наступна:

1. Тестування починається з 1-о процесора. Обираємо першу пару процесорів: 1 та 2. Результат тестування v_{12} = 1. Процесори виключаються з діагностування.
2. Обираємо наступну пару : 3 → 4, v_{34}= 0, будується 0-ланцюжок, ваги P3=1, P4=2.
3. Тестуємо пару 4,5. 4 → 5, v_{45}= 0, пара додається у 0-ланцюжок, ваги P3=1, P4=2, P5=3. Оскільки одна пара виключена з діагностування, можливих відмов може бути t=2, P5>t, отже, 5 процесор – справний.

4. 5 → 6, v_{56}= 0, 6 процесор справний.
5. 5 → 8, v_{58}= 1, 8 процесор несправний.
6. Обираємо для тестування пару {6,7}. Результат тестування «0». 7 процесор справний.
7. 6 → 1, v_{61}= 1, 1 процесор несправний. Результату його тестування на 1-у кроці ми не можем довіряти, тому необхідно провести ще одну перевірку для модуля 2.
8. 7→ 2. v_{72}= 0, отже 2й процесор справний.
9. Залишається один процесор, стан якого невизначений, тестуємо 3й процесор 2м. Результат тестування 1. Процесор 3 несправний. Отже, стан всіх модулей визначений. Діагностування повне.

Стан системи було визначено за n+T-2 = 9 кроків.

4.2. Залежність кількості взаємоперевірок від значень n і t

За допомогою програми самодіагностування, робота якої описана у третьому розділі було проведено ряд тестів для визначення середньої (Таблиця 4.1) та максимальної (Таблиця 4.2) кількості взаємоперевірок процесорів при зміні значень кількості процесорів n та допустимої кількості відмов T. Розглядалися випадки, при яких діагностування було повне. Для порівняння у останньому стовпчику таблиці приведена кількість перевірок при використанні методу. Кількість взаємоперевірок стала – m=2n [3, 7].

Таблиця 4.1 - Середня кількість взаємоперевірок при зміні n та T

n	T=2	T = 3	T = 4	Const=2n
11	8	9	10	22
17	11	13	16	34
56	34	39	43	112

Таблиця 4.2 - Максимальна кількість взаємоперевірок при зміні n та T

n	T=2	T = 3	T = 4	Const=2n
11	11	12	13	22
17	15	18	19	34
56	46	50	53	112

Як видно з даних таблиць, при збільшенні кількості процесорів, кількість взаємоперевірок, необхідних для визначення стану системи, суттєво зменшується, та не досягає верхньої границі, описаної в пункті 4.1.2 даного розділу. При збільшенні допустимого значення T кількість взаємоперевірок також збільшується, оскільки при T=3 та T=4 можуть виникати невизначеності, і необхідно більше перевірок для визначення стану системи.

4.3. Виявлення невизначеностей при використанні запропонованого методу

Під невизначеністю розуміється нездатність методу самодіагностування визначити стан модуля/модулів у системі. В такому випадку стан системи повністю визначити не можна. Наведемо приклад.

Нехай задано діагностичний граф з n=8, T=3, i=1, j=3, несправними процесорами – 1, 2 та 6 (рис.4.3).

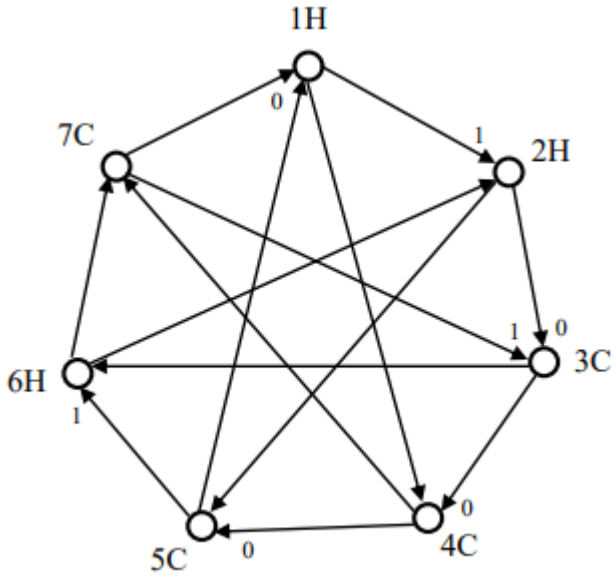


Рисунок 4.3 – Приклад появи невизначеності

Спробуємо визначити стан системи:

1. 1 → 2, v_{12}= 1. Процесори 1 та 2 виключаються з діагностування. Допустима кількість відмов t=T-1=2
2. 3 → 4, v_{34}= 0. Будується 0-ланцюжок, ваги процесорів дорівнюють P3=1, P4=2.
3. 4 → 5, v_{45}= 0. Процесори додаються до 0-ланцюжка. Вага процесора 5 P5>t, тому процесор 5 – справний.
4. 5 → 6, v_{56}= 1. Процесор 6 – несправний. Оскільки t=1, а вага P4=2, процесор 4 – справний.
5. 5 → 1, v_{51}= 1. Процесор 1 – несправний.
6. 4 → 7, v_{47}= 0. Процесор 7 – справний.
7. 7 → 3, v_{73}= 0. Процесор 3 – справний.

Залишився невизначеним лише стан 2-о процесора. Оскільки 2-й процесор тестується двома несправними, його стан визначити не можна, оскільки результати перевірок першим та шостим процесорами не є надійними. Виконуємо перевірки 2-м процесором 3-о та 5-о процесорів. Якщо результати тестів дорівнюватимуть v23=1 або/та v25=1, процесор 2 визначається, як

несправний, оскільки його результати суперечать станам визначених процесорів.

Якщо 2-й процесор тестує 3-й та 5-й процесори правильно, то стан 2-о процесора в такому випадку точно визначити неможливо.

З попереднього прикладу можна зробити певні висновки щодо умов виникнення невизначеностей при діагностуванні багатопроцесорної системи, а саме:

- тестування модуля, стан якого невизначено, двома несправними модулями;
- правильний результат тестування несправним процесором будь-якого іншого модуля, стан якого точно відомий.

Невизначеності можна уникнути, якщо виконується лише одна із зазначених вище умов.

У першому випадку це можливо, якщо невизначений модуль - несправний, та надає неправильний результат тестування процесора, стан якого вже визначено.

У другому випадку невизначеність не виникає, якщо хоча б один модуль, що тестує невизначений модуль, є справним, і його стан можна визначити.

Якщо виконуються обидві умови, і під час діагностування з’являється невизначеність, необхідно ще одна/дві додаткові перевірки одним із справних модулей. Це означає, що в системі, окрім зв’язків діагностичного графа, має бути додаткові зв’язки, які не використовуються під час самодіагностування, але, які можна для усунення невизначеності.

4.4. Особливості аналізу програми самодіагностування

При моделюванні процесу самодіагностування можуть виникати певні труднощі, пов’язані з появою зайвих перевірок та необхідністю покращення аналізу результатів тестування. Наприклад, з’являється питання вибору першої

пари процесорів, з якої має починатись тестування, вибір наступної пари та навіть вибір процесора для перевірки після визначення стану одного із справних процесорів.

Розглянемо особливості роботи програми діагностування та рішення щодо організації процесу самодіагностування, які не були описані в основному алгоритмі у попередньому розділі, але були знайдені в процесі налаштування програми:

- Який процесор з виключеної пари має тестуватися першим, якщо є можливість вибору.

Розглянемо ситуацію, зображену на рисунку 4.4. Параметри системи наступні: n=8, T=3, i=1, j = 3. Один із несправних процесорів – 2.

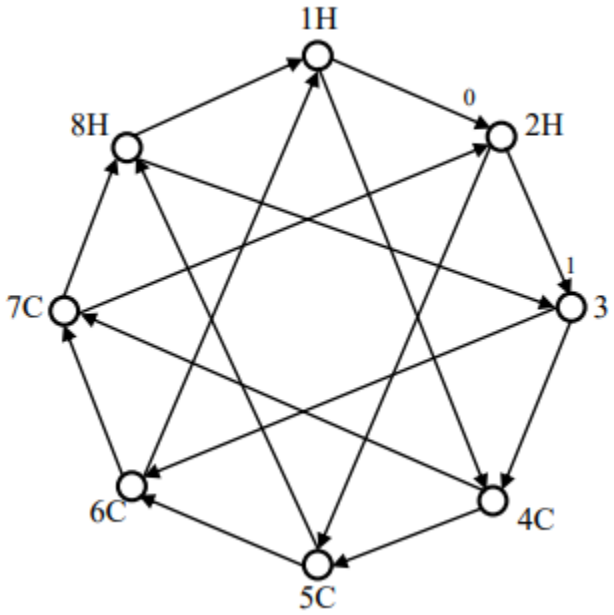


Рисунок 4.4 – Діагностичний граф з параметрами n=8, t=3, несправні процесори – 1, 2, 8

На першій перевірці тестується пара процесорів 1 та 2. Результат тестування «0», будується 0-ланцюжок, обирається наступна пара для тестування. На другій перевірці виключаються з процесу тестування пара процесорів 2 та 3, оскільки результат тестування «1». Далі тестується пара

процесорів 4 та 5, vij=0, тому будується 0-ланцюжок, та обирається наступна пара. Тестуються пара {5, 6}, процесори заносяться у 0-ланцюжок, визначаються ваги процесорів. Оскільки вага P6 більше, ніж T, то 6 процесор справний. Визначаються стани процесорів, які тестуються 6-м. 7-й процесор справний. Є два варіанти, як проводити тестування далі: тестувати 6-м процесором 1-й або провести тестування 7-м процесором 2-й.

При першому варіанті v61 = 1, 1-й процесор - несправний, далі визначається стан 2-о процесора після перевірки v72 = 1 та 8-о перевіркою v78 = 1. Стан системи визначений, діагностування повне.

При другому варіанті одразу проводить перевірка u72, визначається стан 2-о процесора, та одразу визначається стан 1-о процесора. Оскільки нам з першого кроку відомий результат тесту v12 = 0, і цей результат суперечить перевірці v72, то 1-й процесор також є несправним. На останньому кроці визначається стан 8-о процесора. Стан системи визначений.

Тобто при використанні другого варіанту процес самодіагностування потребує на 1 перевірку менше, ніж при першому варіанті. Робиться наступний висновок: якщо існує 3 та більше процесорів (рис.4.5), що пов’язані дугами, результат тестування яких уже відомий, і існує справний процесор i, який пов’язаний з некрайній процесором j у такому ланцюжку, перевірку uij необхідно зробити першою.

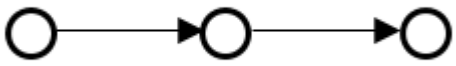


Рисунок 4.5. – Ланцюжок з 3-х процесорів

- Розглянемо декілька окремих випадків, при яких кількість взаємоперевірок процесорів для визначення стану системи є мінімальною

  1.1. Система визначає стан за T+2 перевірки

Нехай початкові параметри діагностичного графу наступні: n = 7, T = 3, i = 1, j = 3, (рис.4.6). Опишемо процес самодіагностування:

1. 1 → 2, v_{12} = 0, будується 0-ланцюжок. Вага процесорів 1 та 2 відповідно.
2. 2 → 3, v_{23} = 0, процесори додаються у 0-ланцюжок. Ваги процесорів дорівнюють P1=1, P2=2, P3=3.
3. 3 → 4, v_{34} = 0, процесори додаються у 0-ланцюжок. Ваги процесорів дорівнюють P1=1, P2=2, P3=3, P4=4. Оскільки P4>T, то 4 процесор справний.
4. 4 → 7, v_{47} = 0, оскільки стан 4-о процесору визначений, його результатам тестування можна довіряти, отже, 7 процесор - справний.
5. 7 → 3, v_{73} = 1, 1 процесор – несправний і несправними є також процесори 1 та 2, оскільки їх результати тестування суперечать стану визначеного процесора. Всі несправні процесори визначено, діагностування повне.

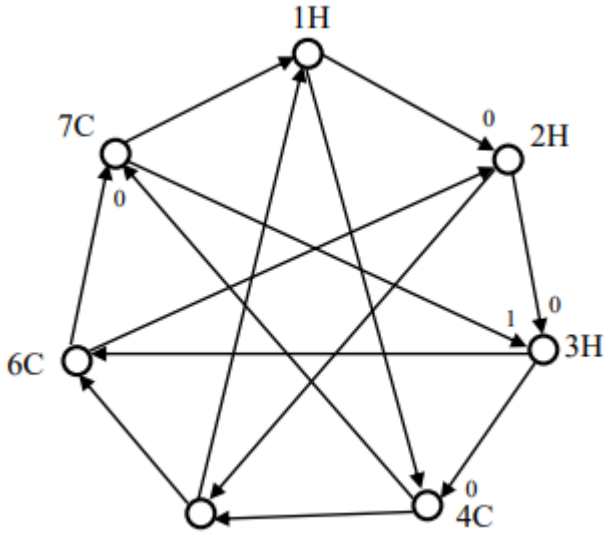


Рисунок 4.6 – Граф для випадку 2.1

- Система визначає стан за *2T* перевірки

Нехай початкові параметри діагностичного графу наступні: *n* = 7, *T* = 3, *i* = 1, *j* = 3, (рис.4.7). Опишемо процес самодіагностування:

1. 1 → 2, $v_{12} = 1$. Процесори виключаються з процесу самодіагностування. Допустима кількість відмов зменшується на 1 і дорівнює тепер 2.
2. 3 → 4, $v_{34} = 1$. Процесори 3 та 4 виключаються з процесу самодіагностування. Допустима кількість відмов -1.
3. 5 → 6, $v_{56} = 1$, Процесори 6 та 5 виключаються з процесу самодіагностування, допустима кількість відмов дорівнює 0, отже, 7 процесор справний.
4. 7 → 1, $v_{71} = 0$, отже, 1 процесор – справний, 2 - несправний.
5. 1 → 4, $v_{14} = 0$, 4 процесор – справний, 3 процесор – несправний.
6. 4 → 5, $v_{45} = 0$, 5 процесор – справний, 6 процесор – несправний, Всі несправні процесори визначено, діагностування повне.

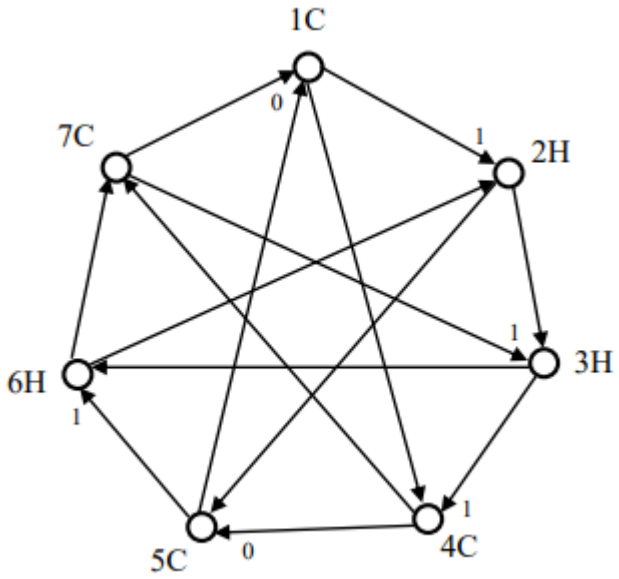


Рисунок 4.7 – Граф для випадку 2.2

- Вибір першої пари процесорів

У всіх прикладах, які були розглянуті в магістерській дисертації, тестування починалось завжди з 1-о процесора. Але це не обов’язкова умова. Першим процесор названий умовно, і при організації самодіагностування у

реальній системі, визначення стану системи при кожній новій перевірці може починатися з довільно обраного процесора.

4.5. Висновки

У даному розділі проводиться аналіз роботи програми самодіагностування, в основі якої закладено метод, запропонований у розділі 3. Виведено залежність кількості взаємоперевірок від зміни параметрів n та T. Розглянуто проблему появи невизначеностей при самодіагностуванні системи, проаналізовано умови їх появи та способи їх усунення.

Розглянуто особливості аналізу програми самодіагностування та рішення щодо організації процесу самодіагностування, які не були описані в основному алгоритм, але були знайдені в процесі налаштування програми.

Знайдено та обґрунтовано значення нижньої та верхньої границь кількості взаємоперевірок, при яких визначається стан системи.

ВИСНОВКИ

В ході виконання магістерської дисертації було розглянуто задачу організації самодіагностування багатопроцесорної системи, проведено аналіз існуючих методів самодіагностування багатопроцесорних систем. Розроблено метод скорочення часу самодіагностування багатопроцесорних систем.

Метод передбачає використання конструктивно регулярних графів для опису діагностичних зв'язків всередині системи. Алгоритм, що реалізує метод, виконує тестування системи з урахуванням особливостей графу діагностичних зв'язків, векторів розміщення справних і несправних процесорів, а також моделі Препарата-Метца-Чена.

Встановлено верхню та нижню границю кількості взаємоперевірок модулів при використання методу. Мінімальна кількість взаємоперевірок дорівнює значенню – $T+1$. Верхня межа кількості тестових перевірок, при якій можна визначити стан системи дорівнює значенню ($n+T$) за винятком випадків, коли з'являються одна або дві невизначеності. Тобто для визначення стану всіх процесорів системи у середньому потребує одну перевірку для справного модуля та 2 - для несправного. В ході роботи було створено програмну реалізацію описаного вище методу. При зміні різних параметрів, а саме: кількості процесорів, максимальної кількості несправних модулів, реальних значень тестування процесорів, максимальна кількість перевірок не перевищувала значення $n+T-2$.

Запропонований метод самодіагностування багатопроцесорних систем дозволяє зменшити загальний час, який витрачає система на визначення власного стану, шляхом зменшення кількості перевірок. Оскільки в кожний момент часу може тестуватися лише одна пара процесорів, то всі інші модулі, які не приймають участі в тестуванні, не припиняють виконання своїх задач.

Подальша робота може бути націлена на збільшення діагностичних зв’язків між процесорами та збільшення допустимої кількості несправних процесорів T.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Blanke, M., Kinnaert, M., Lunze, J., & Staroswiecki, M. (2015). Diagnosis and Fault-tolerant Control, 3rd Edition. (3. edition ed.) Springer. DOI: 10.1007/978-3-662-47943-8.
2. K. Abrougui and M. Elhadef. Parallel Self-Diagnosis of Large Multiprocessor Systems Under the Generalized Comparison Model. In Proc. of the 11th Int. Conf. on Parallel and Distributed Systems, Fukuoka, Japan. (To appear)., Jul. 2005.
3. Романкевич В.О. Методи і засоби оцінки технічних характеристик гарантоздатності відмовостійких багатопроцесорних систем управління складними об’єктами: Дис. на здобуття наукового ступеня доктора технічних наук: 05.13.05; - Захищена 29.01.2018; Затв. 20.03.2018.- К., 2017.- 388с.
4. Prepatata FP, Metze G., Chien RT On the Connection Assignment Problem of Diagnosable Systems. / / IEEE Trans. Electronic Comput. - 1987. - Vol.EC-16, № 6. - P.848-854.
5. Ведешенков В.А., “О диагностировании отказавших модулей в цифровых системах с помощью цепочек из трех модулей”, АиТ, 2000, №8, 156–164 Math-Net.Ru; Vedeshenkov V.A., “On Diagnosis of Failed Modules in Digital Systems by Three-Module Chains”, Autom. Remote Control, 61:8, Part 2 (2000), 1382–1389.
6. Ю. К. Димитриев, Эффективность локального самодиагностирования в вычислительных системах с циркулянтной диагностической структурой, ПДМ, 2008, номер 2, 96–101
7. Romankevich V. A. Self-testing of multiprocessor systems with regular diagnostic connections / V. A. Romankevich // Automation and Remote Control. – 2017. – Vol. 78, Issue 2. – P. 289 – 299.

8. Довганюк А.О. Метод зменшення часу самодіагностування багатопроцесорних систем / Т.Г. Сапсай, Довганюк А.О. // Прикладна математика та комп‘ютінг (ПМК-2018). Збірник тез доповідей. – Київ, НТУУ «КПІ». – 2018.
9. Довганюк А.О. О последовательном диагностировании многопроцессорных систем/ Т.Г. Сапсай, Довганюк А.О. // Системный анализ и информационные технологии: материалы 20-й Международной научно-практической конференции SAIT 2018. – К.: УНК «ИПСА» КПИ им. Игоря Сикорского, 2018. – С. 186.
10. Mark Lutz. Learning Python, Fifth edition. Published by O’Reilly Media, Inc., 2013.
11. Overview of NetworkX [Електронний ресурс] – 2013 – Режим доступу: https://networkx.github.io/documentation/stable/ - Дата доступу: грудень 2018.
12. Эффективность локального само диагностирования в вычислительных системах с циркулянтной диагностической структурой [Електронний ресурс] – 2013 – Режим доступу: https://cyberleninka.ru/article/n/effektivnost–lokalnogo–samo-diagnostirovaniya–v–vychislitelnyh – sistemah – s–tsirkulyantnoy–diagnosticheskoy-strukturoy/ - Дата доступу: грудень 2018.
13. Phillips DJ, McGlaughlin A, Ruth D, Jager LR, Soldan A. Graph theoretic analysis of structural connectivity across the spectrum of Alzheimer’s disease: The importance of graph creation methods. Neuroimage Clin. 2015;7:377–390. doi: 10.1016/j.nicl.2015.01.007.

ДОДАТКИ

Додаток 1. Копії графічних матеріалів

Додаток 2. Код

Додаток 3. Публікації по темі магістерської дисертації